User's Manual

Digital Gamma Finder (DGF)

Pixie-4

Version 2.50, June 2012

XIA LLC

31057 Genstar Road
Hayward, CA 94544 USA

Phone: (510) 401-5760; Fax: (510) 401-5761
http://www.xia.com

Disclaimer

Information furnished by XIA is believed to be accurate and reliable. However, XIA assumes no responsibility for its use, or for any infringement of patents, or other rights of third parties, which may result from its use. No license is granted by implication or otherwise under the patent rights of XIA. XIA reserves the right to change the DGF product, its documentation, and the supporting software without prior notice.

Table of Contents

1 Overview....3
1.1 Features....3
1.2 Specifications....4
2 Setting Up....5
2.1 Installation....5
2.2 Getting Started....6
3 Navigating the Pixie Viewer....10
3.1 Overview....10
3.2 Setup Group....11
3.3 Run Control Group....16
3.4 Results Group....16
3.5 Optimizing Parameters....18
3.6 File Series....20
4 Data Runs and Data Structures....24
4.1 Run Types....24
4.2 Output Data Structures....26
5 Hardware Description....32
5.1 Analog Signal Conditioning....32
5.2 Real-time Processing Units....33
5.3 Digital Signal Processor (DSP)....33
5.4 PCI Interface....34
6 Theory of Operation....35
6.1 Digital Filters for γ-ray Detectors....35
6.2 Trapezoidal Filtering in the Pixie-4....37
6.3 Baselines and Preamplifier Decay Times....38
6.4 Thresholds and Pile-up Inspection....39
6.5 Filter Range....42
6.6 Dead Time and Run Statistics....42
7 Operating Multiple Pixie-4 Modules Synchronously....55
7.1 Clock Distribution....55
7.2 Trigger Distribution....57
7.3 Run Synchronization....60
7.4 External Gate and Veto (GFLT)....61
7.5 External Status....63
7.6 Coincident Events....64
8 Using Pixie-4 Modules with Clover detectors....68
9 Troubleshooting....69
9.1 Startup Problems....69
9.2 Acquisition Problems....71
10 Appendix A....73
10.1 Front end jumpers for termination and attenuation....73
10.2 Clock Jumpers....74
10.3 PXI backplane pin functions....75
10.4 Control and Status Register Bits....76

© XIA 2012. All rights reserved.

1 Overview

The Digital Gamma Finder (DGF) family of digital pulse processors features unique capabilities for measuring both the amplitude and shape of pulses in nuclear spectroscopy applications. The DGF architecture was originally developed for use with arrays of multi-segmented HPGe gamma-ray detectors, but has since been applied to an ever broadening range of applications.

The DGF Pixie-4 is a 4-channel all-digital waveform acquisition and spectrometer card based on the CompactPCI/PXI standard for fast data readout to the host. It combines spectroscopy with waveform capture and on-line pulse shape analysis. The Pixie-4 accepts signals from virtually any radiation detector. Incoming signals are digitized by 14-bit 75 MSPS ADCs. Waveforms of up to 13.6 μs in length for each event can be stored in a FIFO. The waveforms are available for onboard pulse shape analysis, which can be customized by adding user functions to the core processing software. Waveforms, timestamps, and the results of the pulse shape analysis can be read out by the host system for further off-line processing. Pulse heights are calculated to 16-bit precision and can be binned into spectra with up to 32K channels. The Pixie-4 supports coincidence spectroscopy and can recognize complex hit patterns.

Data readout rates through the CompactPCI/PXI backplane to the host computer can be over 100Mbytes/s. The PXI backplane is also used to distribute clocks and trigger signals between several Pixie-4 modules for group operation. With a large variety of CompactPCI/PXI processor, controller or I/O modules being commercially available, complete data acquisition and processing systems can be built in a small form factor.

1.1 Features

- Designed for high precision γ-ray spectroscopy with HPGe detectors.
- Directly compatible with scintillator/PMT combinations: NaI, CsI, BGO, and many others.
- Simultaneous amplitude measurement and pulse shape analysis for each channel.
- Input signal decay time: as fast as 150ns and up to 10ms, exponentially decaying.
- Wide range of filter rise times: from 53ns to 109μs, equivalent to 27ns to 50μs shaping times.
- Programmable gain and input offset.
- Excellent pileup inspection: double pulse resolution of 50 ns. Programmable pileup inspection criteria include trigger filter parameters, threshold, and rejection criteria.
- Digital oscilloscope and FFT for health-of-system analysis.
- Triggered synchronous waveform acquisition across channels, modules and crates.
- Dead times as low as 1 μs per event are achievable (limited by DSP algorithm complexity). Events at even shorter time intervals can be extracted via off-line ADC waveform analysis.
- Digital constant fraction algorithm measures event arrival times down to a few ns accuracy.
- Supports 32-bit 33 MHz PCI data transfers (>100 Mbytes/second).

© XIA 2012. All rights reserved.

1.2 Specifications

Front Panel I/O	
Signal Input (4x)	4 analog inputs. Selectable input impedance: 50Ω and 5kΩ, ±5V pulsed, ±2V DC. Selectable input attenuation 1:7.5 and 1:1 for 50Ω setting.
Logic Input/Output	General Purpose I/O connected to programmable logic (Rev. C, D only). Currently can be either used as input for global backplane signals Veto or Status, as an input for module specific logic level reported in the data stream, or as an external trigger.
Logic Output	General Purpose output from Digital Signal Processor. Function to be determined.
Backplane I/O	
Clock Input/Output	Distributed 37.5 MHz clock on PXI backplane.
Triggers	Two wired-or trigger buses on PXI backplane. One for synchronous waveform acquisition, one for event triggers.
Status	Global logic level from backplane reported for each event
Token	Global logic level from backplane used for coincidence tests
Synch	Wired-or SYNC signal distributed through PXI backplane to synchronize timers and run start/stop to 50ns.
Veto	Global logic level to suppress event triggering.
Channel Gate	Individual GATE to suppress event triggering for each channel with use of PXI-PDM (Rev. D only)
Data Interface	
PCI	32-bit, 33MHz Read/Write, memory readout rate to host over 100 Mbytes/s.
Digital Controls	
Gain	Analog switched gain from 0.97 to 11.25 in max. 10% steps. Digital gain adjustment of up to ±10% in 15ppm steps.
Offset	DC offset adjustment from –2.5V to +2.5V, in 65535 steps.
Shaping	Digital trapezoidal filter. Rise time and flat top set independently: 53ns – 109μs in small steps.
Trigger	Digital trapezoidal trigger filter with adjustable threshold. Rise time and flat top set independently from 26ns to 853ns.
Data Outputs	
Spectrum	1024-32768 channels, 32 bit deep (4.2 billion counts/bin). Additional memory for sum spectrum for clover detectors.
Statistics	Real time, live time, filter and gate dead time, input and throughput counts.
Event data	Pulse height (energy), timestamps, pulse shape analysis results, waveform data and ancillary data like hit patterns.

© XIA 2012. All rights reserved.

2 Setting Up

2.1 Installation

2.1.1 Hardware Setup

The Pixie-4 modules can be operated in any standard 3U CompactPCI or PXI chassis. Chassis following only the CompactPCI standard can be used to operate modules individually. To operate several modules together, a backplane following the PXI standard must be present. Put the host computer (or remote controller) in the system slot of your chassis. Place the Pixie-4 modules into any free slots with the chassis still powered down, then power up the chassis (Pixie-4 modules are not hot swappable). If using a remote controller, be sure to boot the host computer *after* powering up the chassis[1].

2.1.2 Drivers and Software

System Requirements: The Pixie software is compatible with Windows XP, Vista, or Windows 7. Restrictions apply to the 64 bit version of Windows 7[2]. Please contact XIA for details on operating Pixie-4 modules with Linux.

When the host computer is powered up the first time after installing the controller and Pixie-4 modules in the chassis, it will detect new hardware and try to find drivers for it. (A Pixie-4 module will be detected as a new device every time it is installed in a new slot.) While there is no required order of installation of the driver software, the following sequence is recommended (users with embedded host computer skip to step 4):

1. If you have a remote controller, first install the driver software for the controller itself. Otherwise, skip to step 4.
   Unless directed otherwise by the manufacturer of the controller, this can be done with or without the controller and Pixie-4 modules installed in the host computer and/or chassis. If the modules are installed, ignore attempts by Windows to install drivers until step 7.

   NI controllers come with a multi-CD package called "Device Driver Reference CD". For simplicity it is recommended to install the software on these CDs in the default configuration.
2. Unless already installed, power down the host computer, install the controller in both the host computer and chassis, and power up the system again (chassis first).
3. Windows will detect new hardware (the controller) and should find the drivers automatically. Verify in Window's device manager that the controller is properly installed and has no "resource conflicts".
4. Install Igor Pro

[1] In some systems, "scan for hardware changes" in the Windows device manager may detect and install a remote chassis when the PC was booted first.

[2] At the time of writing, these restrictions are: National Instrument's MXI-3 and MXI-4 controllers do not support Windows 7 (64 bit) on systems with more than 4 GB RAM. Embedded PC controllers and the "MXI Express" series of controller bridges appear to be fully compatible with Windows 7 (64 bit).

© XIA 2012. All rights reserved.

5. Install the Pixie-4 software provided by XIA (see section 2.2.3)
6. Unless already installed, power down the host computer and install the Pixie-4 modules in the chassis. Check the input jumper settings for the appropriate signal termination: 50 Ω or 5 kΩ (see section 10.1 for details). Then power up the system again (chassis first).
7. Windows will detect new hardware (the Pixie-4 modules) and should find the drivers automatically. If not, direct it to the "drivers" directory in the Pixie-4 software distribution installed in step 5. Verify in Window's device manager that the modules are properly installed as "PLX Custom (OEM) PCI 9054 Boards (32)" or "... (64)" and have no "resource conflicts". Currently, the driver must be version 6.5.0.[3] The previously used driver version 4.1 will identify the modules as "Custom (OEM) PCI 9054 Boards" without the "PLX"

2.1.3 Pixie User Interface

The Pixie Viewer, XIA's graphical user interface to set up and run the Pixie-4 modules, is based on WaveMetrics' IGOR Pro. To run the Pixie Viewer, you have to have IGOR Version 5.0 or higher installed on your computer. By default, IGOR Pro will be installed at C:\Program Files\WaveMetrics\IGOR Pro Folder.

The CD-ROM with the Pixie-4 software distribution contains

1. an installation program Setup.exe,
2. the Pixie-4 software in the folder XIA\Pixie4 and its subfolders.

The Pixie-4 software can be installed by running its installation program. Follow the instructions shown on the screen to install the software to the default folder selected by the installation program, or to a custom folder. This folder will contain the IGOR control program (Pixie4.pxp), online help files and 8 subfolders (Configuration, Doc, Drivers, DSP, Firmware, MCA, PixieClib, and PulseShape). Make sure you keep this folder organization intact, as the IGOR program and future updates rely on this. Feel free, however, to add folders and subfolders at your convenience.

For the latest version of the Pixie Viewer software, go to support.xia.com and search for "Pixie release".

2.2 Getting Started

To start the Pixie Viewer, double-click on the file "Pixie4.pxp" in the installation folder. After IGOR loaded the Pixie Viewer, the START UP[4] panel should be prominently displayed in the middle of the desktop.

In the panel, first select the chassis type and number of Pixie-4 modules in the system. Then specify the slot number in which each module resides.

Click on the *Start Up System* button to initialize the modules. This will download DSP code and FPGA configuration to the modules, as well as the module parameters. If you see messages

[3] For information on using the older PLX drivers (version 6.3.1) with Windows 2000, see the "readme" file in the *Drivers* folder of the software distribution.

[4] In the following, SMALL CAPS are used for panel names; *italic* font is used for buttons and controls.

© XIA 2012. All rights reserved.

similar to "Module 0 in slot 5 started up successfully!" in the IGOR history window, the Pixie-4 modules have been initialized successfully. Otherwise, refer to the troubleshooting section for possible solutions. If you want to try the software without a chassis or modules attached, click on *Offline Analysis*.

Figure 2.1: The Pixie-4 START UP panel (above) and MAIN Panel (right)

After the system is initialized successfully, you will see the MAIN control panel that serves as a shortcut to the most common actions and from which all other panels are called. Its controls are organized in three groups: Setup, Run Control, and Results.

In the Setup group, the *Start System* button opens the START UP panel in case you need to reboot the modules. The *Open Panels* popup menu leads to four panels where parameters and acquisition options are entered. They are described in more detail in section 3 and in the online help. To get started, select *Parameter Setup*, which will open (or bring to front) the PARAMETER SETUP panel shown in Figure 2.2. For most of the actions the Pixie Viewer interacts with one Pixie module at a time. The number of that module is displayed at the top of the MAIN panel and the top right of the PARAMETER SETUP panel. Proceed with the steps below to configure your system.

Note: The *More/Less* button next to the *Help* button on the bottom of the PARAMETER SETUP panel can be used to hide some controls. This may be helpful to first-time Pixie users who only want to focus on the most essential settings.

For an initial setup, go through the following steps:

© XIA 2012. All rights reserved.

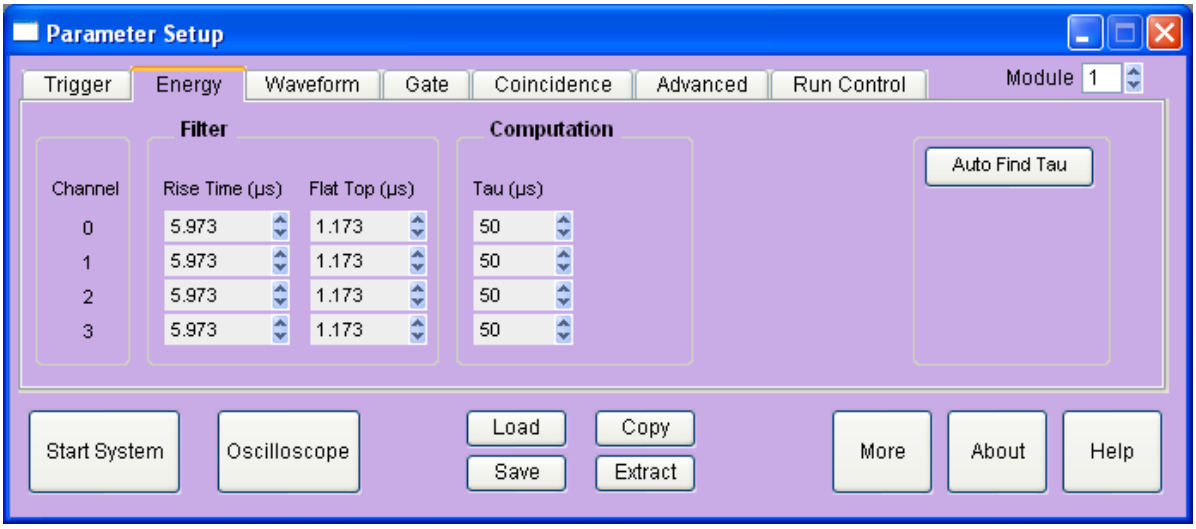


Figure 2.2: The PARAMETER SETUP Panel, *Energy* tab shown

1. If not already visible, open the PARAMETER SETUP panel by selecting *Parameter Setup* from the *Open Panel* popup menu in the MAIN panel.
2. At the bottom of the PARAMETER SETUP panel, click on the *Oscilloscope* button. This opens a graph that shows the untriggered signal input.
   In the OSCILLOSCOPE panel, click *Refresh* to update the display. The pulses should fall in the display range (0-16K). If no pulses are visible or if they are cut off at the upper or lower range of the display, click *Adjust Offsets* to automatically set the DC offset. If the pulse amplitude is too large to fall in the display range, decrease the *Gain*. If the pulses are negative, toggle the *Invert* checkbox.

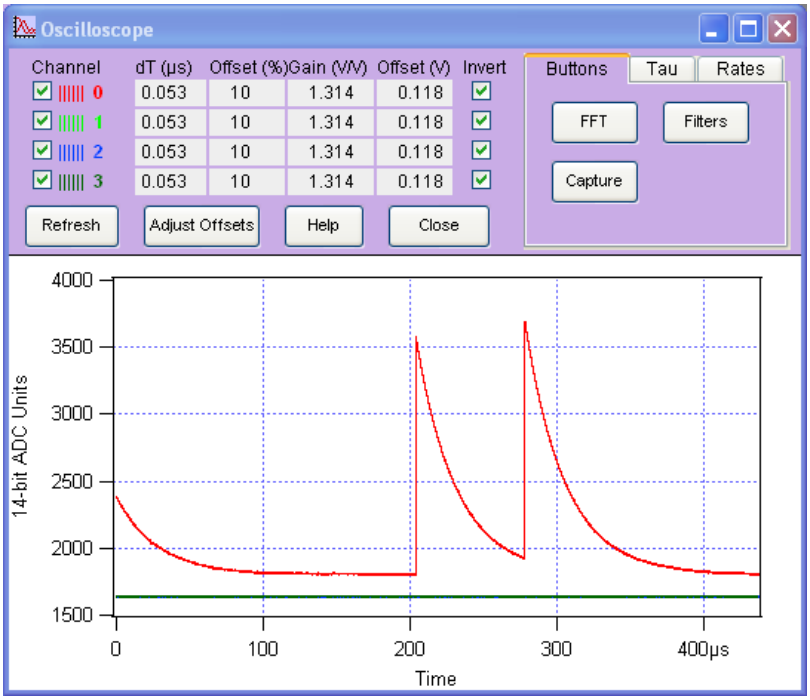


Figure 2.3: **OSCILLOSCOPE panel with typical pulses from a pulser.**

© XIA 2012. All rights reserved.

3. In the *Energy* tab of the PARAMETER SETUP panel, input an estimated preamplifier exponential RC decay time for *Tau*, and then click on *Auto Find Tau* to determine the actual Tau value for all channels of the current module. You can also enter a known good Tau value directly in the *Tau* control field, or use the controls in the OSCILLOSCOPE to manually fit Tau for a pulse.
4. Save the modified parameter settings to file. To do so, click on the *Save* button at the bottom of the PARAMETER SETUP panel to open a save file dialog. Create a new file name to avoid overwriting the default settings file.
5. Save the Igor experiment using *File -> Save Experiment As* from the top menu. This saves the current state of the interface with all open panels and the settings for file paths and slot numbers (the settings independent of module parameters).
6. Click on the *Run Control* tab, set *Run Type* to "0x301 MCA Mode", *Poll time* to 1 second, and *Run time* to 30 seconds or so, then click on the *Start Run* button. A spinning wheel will appear occasionally in the lower left corner of the screen as long as the system is waiting for the run to finish. If you click the *Update* button in the MAIN panel, the count rates displayed in the Results group are updated.
7. After the run is complete, select *MCA Spectrum* from the *Open Panels* popup menu in the Results group of the MAIN panel. The MCA SPECTRUM graph shows the MCA histograms for all four channels. You can deselect other channels while working on only one channel. After defining a range in the spectrum with the cursors and setting the fit option to *fit peaks between cursors*, you can apply a Gauss fit to the spectrum by selecting the channels to be fit in the *Fit* popup menu. You can alternatively enter the fit limits using the *Min* and *Max* fields in the table or by specifying a *Range* around the tallest peak or the peak with the highest energy. To scale the spectrum in keV, enter the appropriate ratio in the field *keV/bin*.

At this stage, you may not be able to get a spectrum with good energy resolutions. You may need to adjust some settings such as energy filter rise time and flat top as described in section 3.5.

© XIA 2012. All rights reserved.

3 Navigating the Pixie Viewer

3.1 Overview

The Pixie Viewer consists of a number of graphs and control panels, linked together by the MAIN control panel. The Viewer comes up in exactly the same state as it was when last saved to file using *File->Save Experiment*. This preserves settings such as the file paths and the slot numbers entered in the START UP panel. However, the Pixie module itself loses all programming when it is switched off. When the Pixie module is switched on again, all programmable components need code and configuration files to be downloaded to the module. Clicking on the *Start Up System* button in the START UP panel performs this download. Below we describe the concepts and principles of using the Pixie Viewer. Detailed information on the individual controls can be found in the Online Help for each panel. The operating concepts are described in sections 4-7 .

The controls in the MAIN control panel are organized in three groups: Setup, Run Control, and Results. In the Setup and Results groups, popup menus lead to the panels and graphs indicated in Figure 3.1.

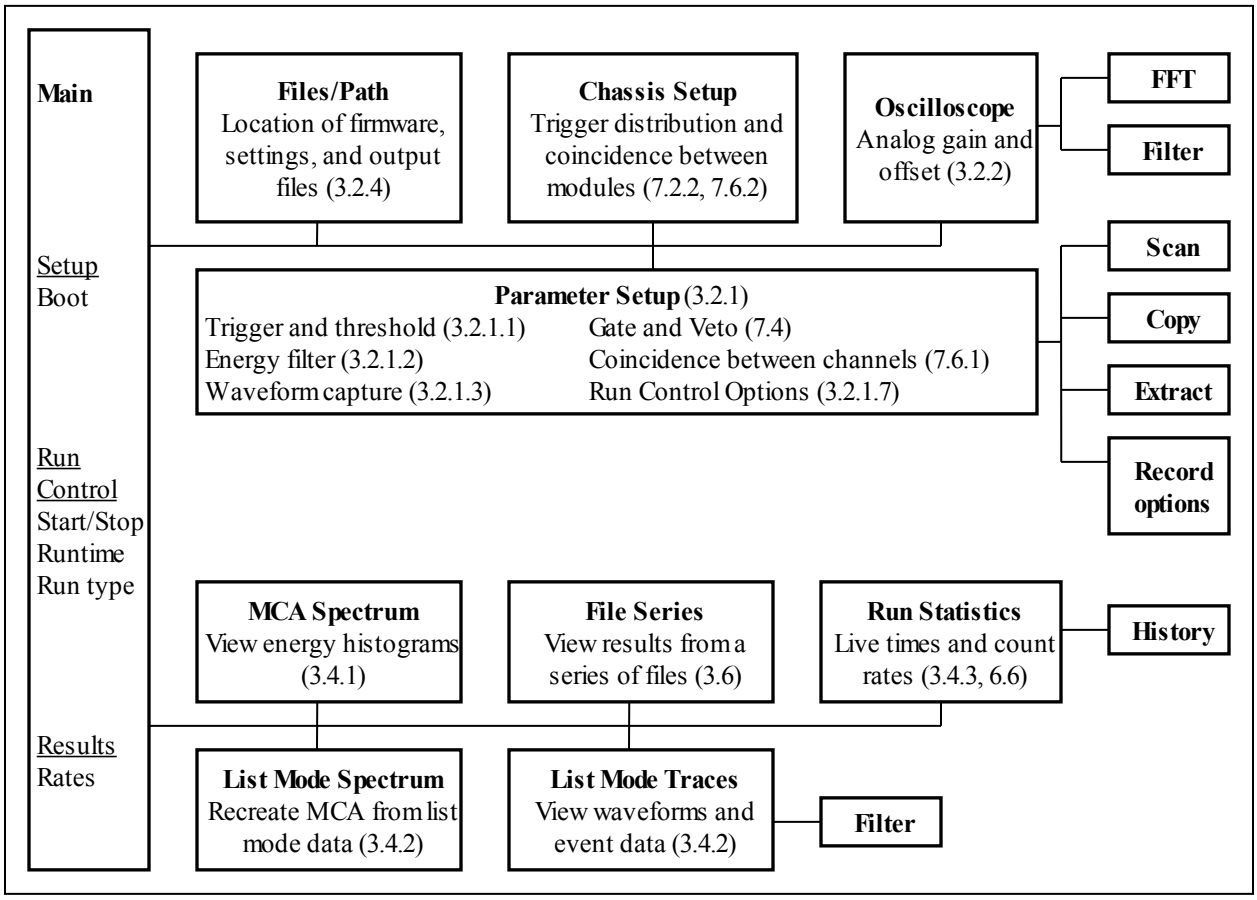


Figure 3.1: Block diagram of the major panels in the Pixie Viewer. Numbers in brackets point to the corresponding section in the user manual. All panels are described in detail in the online help.

© XIA 2012. All rights reserved.

3.2 Setup Group

In the setup group, there is a button to open the START UP panel, which is used to boot the modules. The *Open Panels* popup menu leads to one of the following panels: PARAMETER SETUP, OSCILLOSCOPE, CHASSIS SETUP, FILES/PATHS

3.2.1 PARAMETER SETUP Panel

The PARAMETER SETUP panel is divided into 7 tabs, summarized below. Settings for all four channels of a module are shown in the same tab. At the upper right is a control to select the module to address. At the bottom of the panel is a *More* button, which will make all advanced panel controls visible as well.

The Pixie-4 being a digital system, all parameter settings are stored in a settings file. This file is separate from the Igor experiment file, to allow saving and restoring different settings for different detectors and applications. Parameter files are saved and loaded with the corresponding buttons at the bottom of the PARAMETER SETUP panel. After loading a settings file, the settings are automatically downloaded to the module. At module initialization, the settings are automatically read and applied to the Pixie module from the last saved settings file.

In addition there are buttons to copy settings between channels and modules, and to extract settings from a settings file. Two large buttons at the lower left duplicate the buttons to call the START UP panel and the OSCILLOSCOPE.

3.2.1.1 Trigger Tab

The *Trigger* tab contains controls to set the trigger filter parameters and the trigger *threshold*, together with checkboxes to enable or disable trigger, to control trigger distribution (see section 7.2.1), and to set time stamping options for each channel. Except for the threshold, the trigger settings have rarely to be changed from their default values.

The threshold value corresponds to ¼ of the pulse height in ADC steps, e.g. with a threshold of 20, triggers are issued for pulses above 80 ADC steps. This relation is true if the trigger filter *rise time* is large compared to the pulse rise time and small compared to the pulse decay time. A pulse shape not meeting these conditions has the effect of raising the effective threshold. For a modeled behavior of the trigger, you can open displays from the OSCILLOSCOPE and the LIST MODE TRACES panels that show trigger filter and threshold computed from acquired waveforms using the current settings. The threshold value is scaled with the trigger filter *rise time*, therefore it is not limited to integer numbers.

© XIA 2012. All rights reserved.

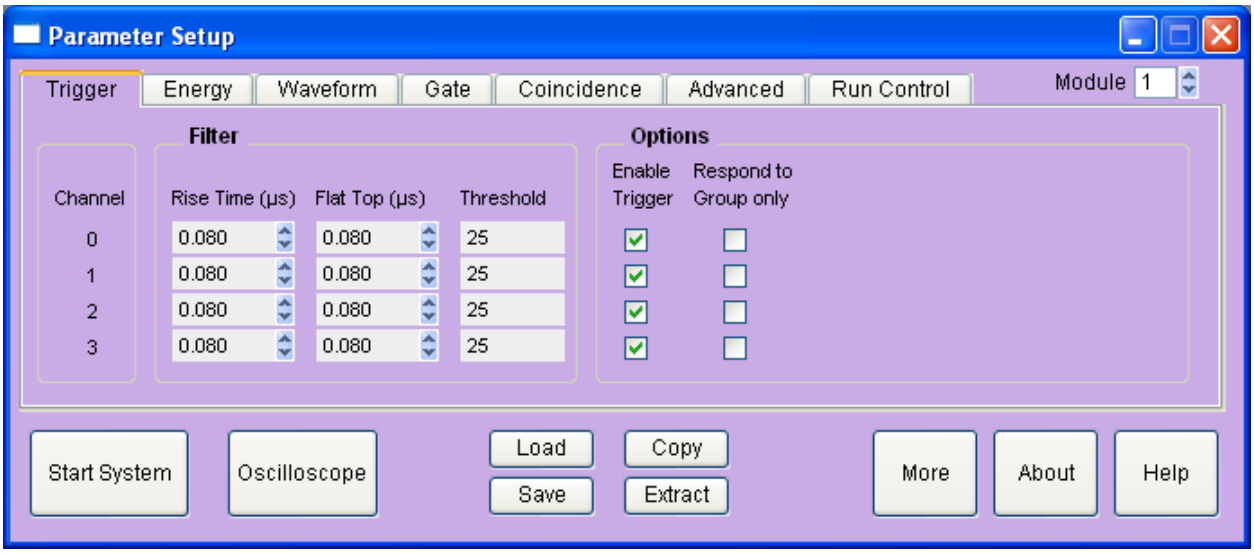


Figure 3.2: The *Trigger* tab of the PARAMETER SETUP panel.

3.2.1.2 Energy Tab

The E*nergy* tab contains the settings for the energy filter and the subsequent computation. These settings are most important for obtaining the best possible energy resolution with a Pixie-4 system. The energy filter *rise time* (or peaking time) essentially sets the tradeoff between throughput and resolution: longer filter *rise times* generally improve the resolution (up to a certain optimum) but reduce the throughput because more time is required to measure each pulse. The pulse decay time *Tau* is used to compensate for the decay of a previous pulse in the computation of the pulse height. You can enter a known good value, or click on *Auto Find Tau* to let the Pixie-4 determine the best value.

The advanced controls in this tab contain functions to modify the energy computation and to acquire a series of measurements with varying filter settings and decay times to find the best settings. For a detailed description of the filter operation, see section 6.

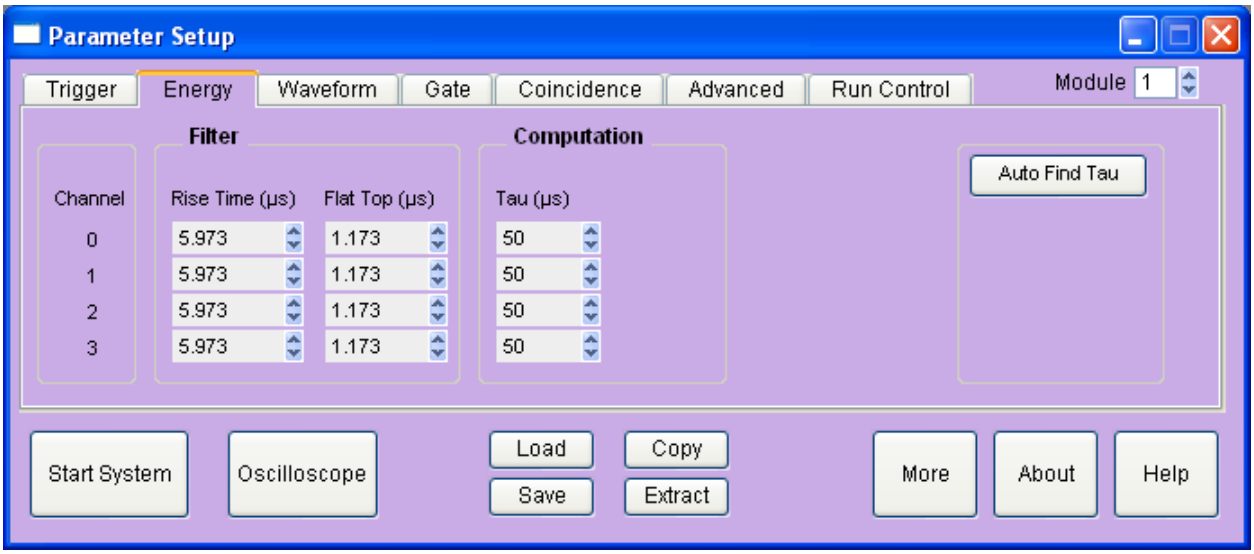


Figure 3.3: The *Energy* tab of the PARAMETER SETUP Panel.

© XIA 2012. All rights reserved.

3.2.1.3 Waveform Tab

The *Waveform* tab contains the controls to set the length and pre-trigger delay of the waveforms to be acquired. Advanced options include parameters for online pulse shape analysis

3.2.1.4 Gate Tab

The *Gate* tab contains the controls to set the window for gating acquisition with external signals. We distinguish

- GATE, a dedicated signal for each individual channel. It is active for the rising edge of the pulse e.g. to suppress a detector pulse with a coincident pulse from a BGO shield.
- VETO, a signal distributed to all modules and channels, but each channel is individually enabled to require or ignore this signal. VETO is active during the validation of a pulse (after pileup inspection), an energy filter rise time plus flat top after the rising edge. With suitable external logic, the decision to veto a pulse can be made from information obtained at the rising edge of the pulse (e.g. multiplicity from several channels) and therefore this function is also called Global First Level Trigger (GFLT).

For a detailed description of the GATE and VETO operation, see section 7.4.

3.2.1.5 Coincidence Tab

The *Coincidence* tab contains the controls to set the acceptable hit pattern, and the coincidence window after validation during which channels can contribute to the hit pattern. There is a checkbox for each possible hit pattern. For example, if the checkbox with pattern 0100 is checked, events with a hit in channel 2 and no others are accepted. Selecting multiple checkboxes accepts combinations of hit patterns, e.g. any event with exactly one channel hit.

For a detailed description of the coincidence operation, see section 7.2.1. Controls for coincidences between modules are located in the CHASSIS SETUP Panel and described in section 7.2.2.

3.2.1.6 Advanced Tab

The A*dvanced* tab contains the controls for modifying the pileup inspection, histogram accumulation, and baseline measurements.

3.2.1.7 Run Control Tab

The *Run Control* tab defines the settings for data acquisition. The “Run Type” popup menu selects MCA or list mode runs, see section 4 for a detailed description. In addition, there are controls

- to set the run time (length of data acquisition as measured by Igor),
- to set the polling time (period for checking if list mode data is available for readout and/or run time is reached),
- to specify the data file name (a *base name* plus 4-digit *run number* that can be made to increment automatically), and

© XIA 2012. All rights reserved.

- to specify the number of spills in list mode runs. (In list mode runs, data is accumulated in on-board memory until full, at which time it is read out by the host PC. We call each such readout a spill. The number of spills thus sets the amount of data to collect.)

The *Start Run* and *Stop Run* buttons from the MAIN control panel are duplicated here as well.

Advanced options include settings for synchronizing acquisition between modules, controls to set a timeout for each spill, the number of events per spill, and the spill readout mode, and a button to open a panel with advance record options.

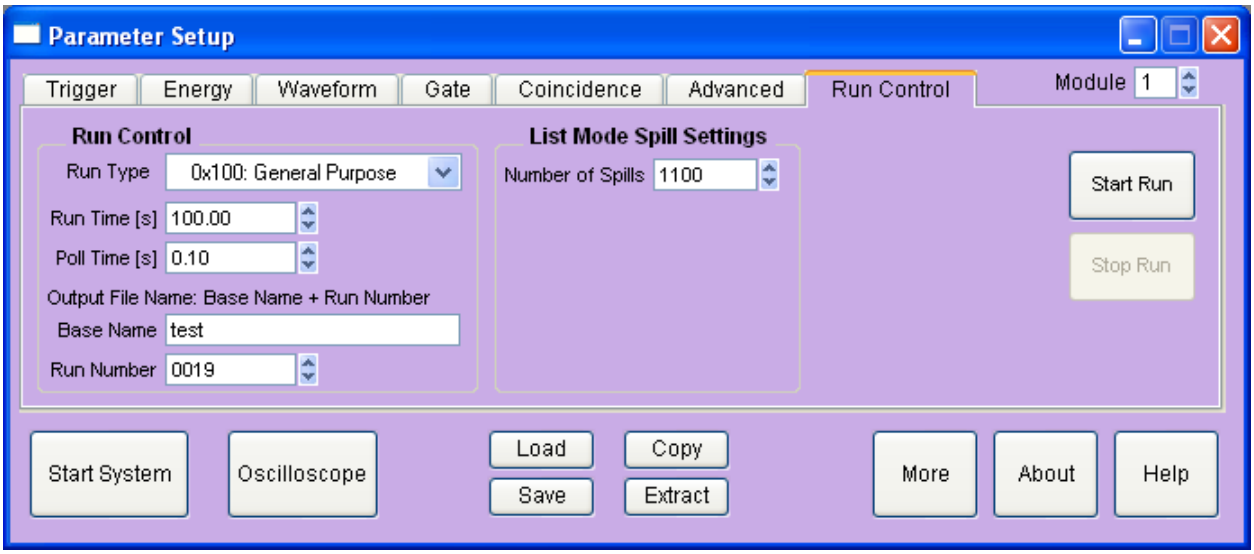


Figure 3.4: The *Run Control* tab of the PARAMETER SETUP Panel.

3.2.2 OSCILLOSCOPE

As mentioned in section 2.3, the OSCILLOSCOPE (Figure 2.3) is used to view untriggered traces as they appear at the ADC input and to set all parameters relating to the analog gain and offset. There are controls titled

- *dT [us],* which sets the time between samples in the oscilloscope (there are always 8192 samples in the oscilloscope window),
- *Offset [%],* which sets the target DC-offset level for automatic adjustment,
- *Gain (V/V),* which sets the analog gain before digitization, and
- *Offset (V),* which directly sets the offset voltage.

The traces from different channels are not acquired synchronously but one after the other. Therefore even if coincident signals are connected to the Pixie-4 inputs, the OSCILLOSCOPE will show unrelated pulses for each channel.

There are also buttons and controls to

- open a display of the FFT of the input signal, which is useful to detect noise sources
- open a display of the waveforms of the trigger filter and energy filter computed from the traces in the oscilloscope
- repeat the action of the *Refresh* button until a pulse is captured. This is useful for low count rates.
- *Fit* the pulses in the OSCILLOSCOPE with an exponential decay function to determine the decay time Tau, and to *accept* the fit value for the module settings.

© XIA 2012. All rights reserved.

- View the current input count rate and the current fraction of time the signal is out of range. These values are updated in the DSP every ~2-3ms if a run is in progress or not. Their precision is in the order of 5-10%, or 50 cps.

3.2.3 CHASSIS SETUP

The CHASSIS SETUP panel is used to set parameters that affect the system as a whole. Examples are trigger distribution between modules, coincidence settings between modules, and the operation of the Pixie-4's front panel input. See sections 7.2.2 and 7.6.2 for details.

3.2.4 FILES/PATHS

The firmware files, DSP files and settings files are defined in the FILES/PATHS panel. Changes will take effect at the next reboot, e.g. when clicking the *Start Up System* button in this panel or in the START UP panel. There is also a button to set the files and paths to the default, relative to the "home path" of the file Pixie.pxp.

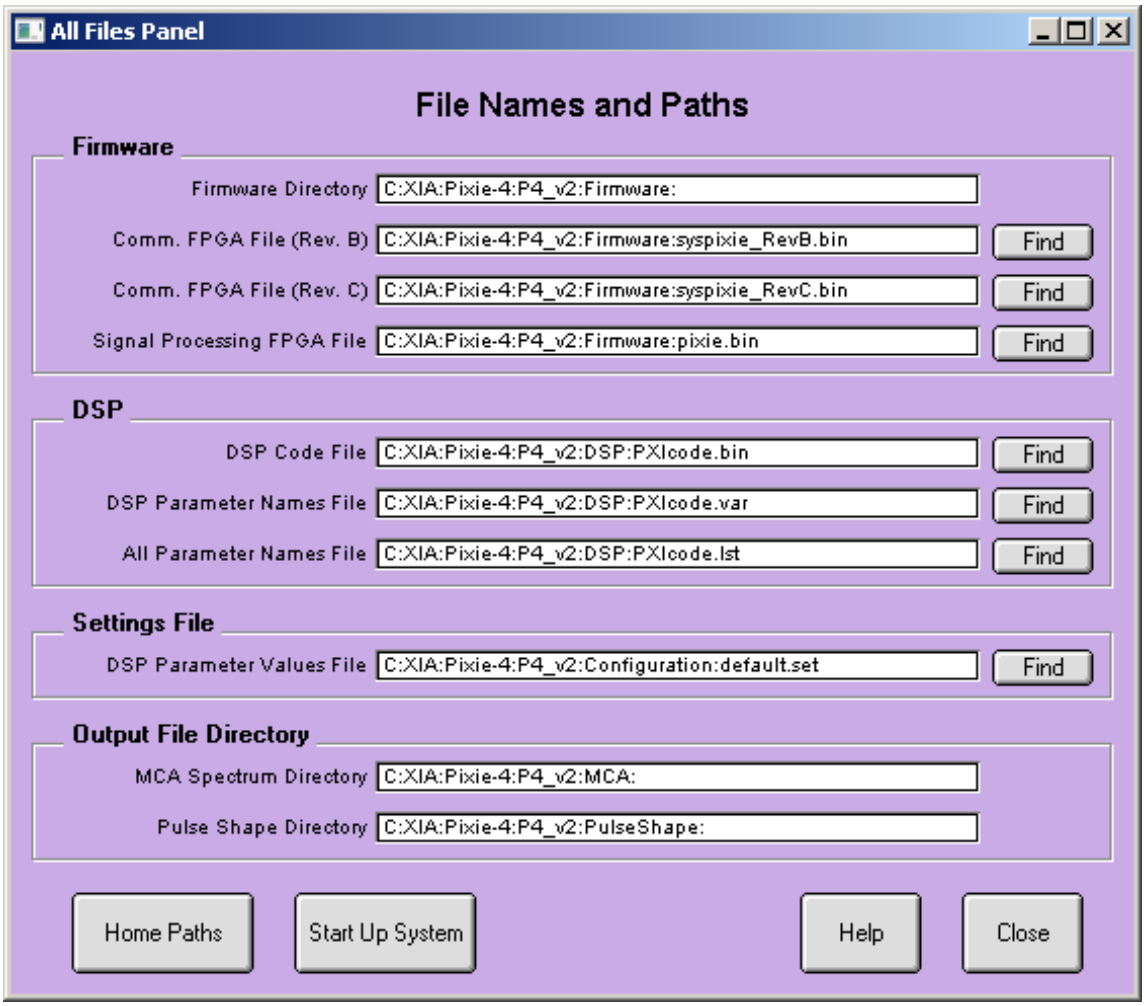


Figure 3.5: The FILES/PATHS Panel.

© XIA 2012. All rights reserved.

3.3 Run Control Group

The Run Control group in the MAIN control panel has the most essential controls to start and stop runs, and to define or monitor the run time and the number of spills. For more options, use the *Run Control* tab of the PARAMETER SETUP panel.

3.4 Results Group

The Results group of the MAIN control panel displays the count rates of the current or most recent run. Click *Update* to refresh these numbers.

The popup menu *Open Panels* leads to panels to view the output data from the data acquisition in detail. These panels are the MCA SPECTRUM display, the LIST MODE TRACES display, the LIST MODE SPECTRUM display, the RUN STATISTICS, and a panel to display results from a series of files.

3.4.1 MCA SPECTRUM

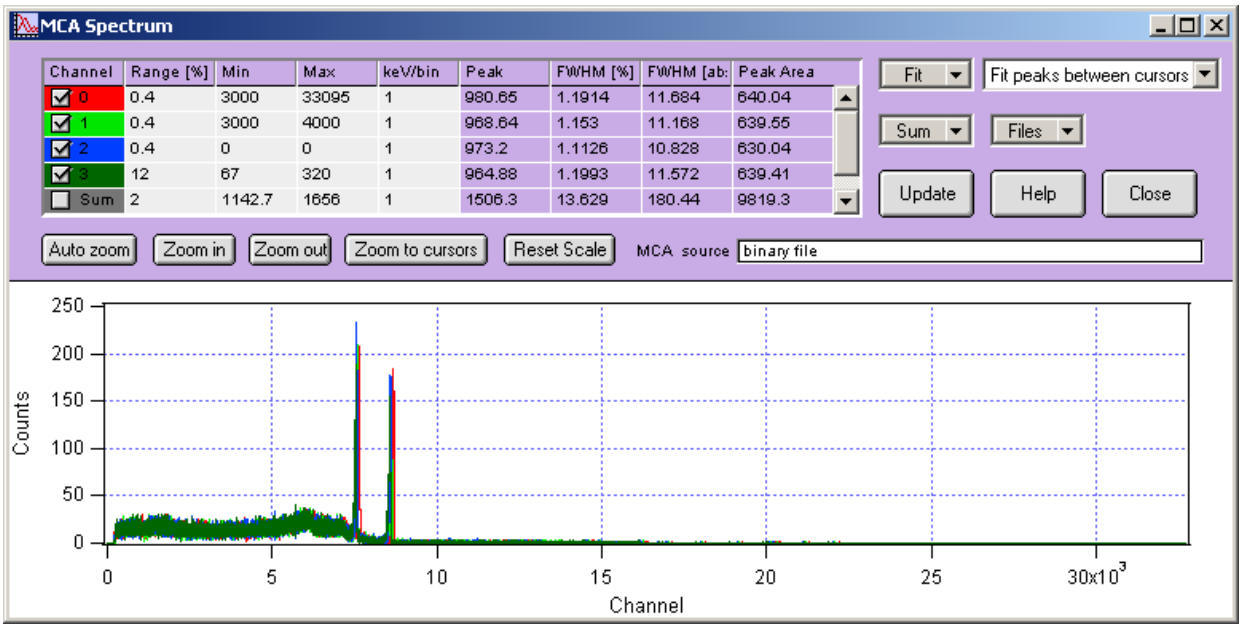


Figure 3.6: The MCA SPECTRUM display.

The MCA SPECTRUM display shows the spectra accumulated in on-board memory or from a .mca file saved at the end of a run. Spectrum analysis is limited to fitting peaks with a Gaussian and computing the peak resolution. There are several options to define the fit range, as described in the online help. Spectra can be saved as text files for import into other applications.

© XIA 2012. All rights reserved.

3.4.2 List Mode Traces and List Mode Spectrum

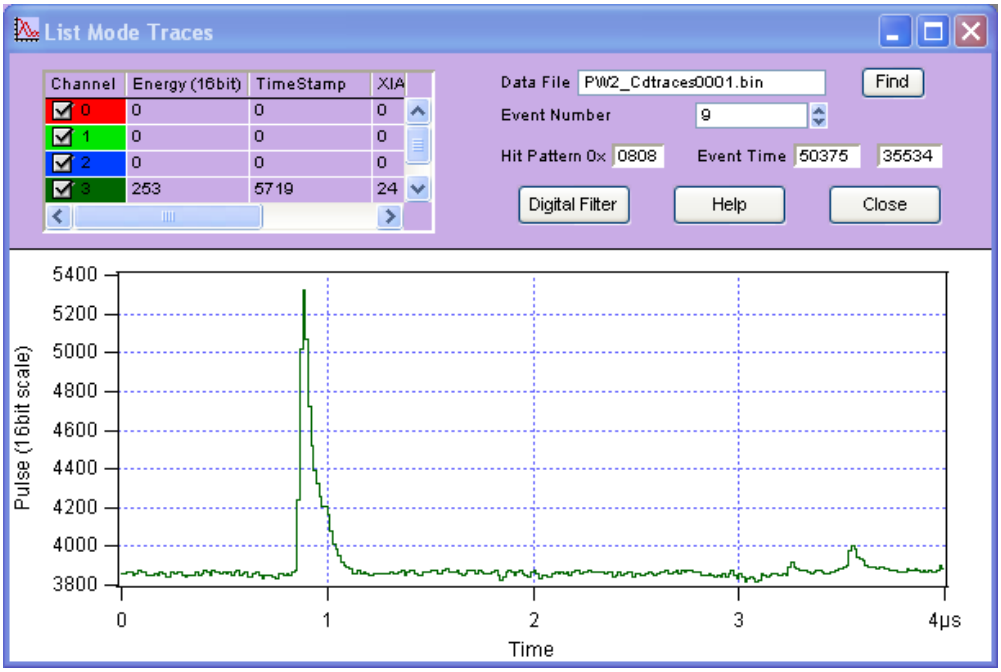


Figure 3.7: The List Mode Traces display.

The List Mode Traces display shows the data from the binary list mode files (.bin). If waveforms were collected, they are shown in the graph section of the panel. Event and channel header information – energy, time stamps, and hit patterns as described in section 4.1.2 – are shown in the fields above the graph section. After specifying a data file with the *Find* button, you can select an event to view by entering its number in the *Event Number* field.

The button *Digital Filters* opens a new plot that shows the response of the trigger filter and energy filter computed from the list mode waveforms. This plot is more precise than the related graph opened from the Oscilloscope since it uses the same full rate (13.3ns) data as the filters implemented in the module, not the reduced rate sampled at the oscilloscope's *dT*. However, unless long list mode traces are acquired or energy filters are short, there may not be sufficient data to compute the energy filter properly.

The List Mode Spectrum display is a plot similar to the MCA Spectrum, but it is computed from the energies saved in the list mode data file. Since energies are stored there in full 16 bit precision, binning can be made finer than in the MCA Spectrum, which is limited to 32K bins. See the online help for a detailed description of the controls.

© XIA 2012. All rights reserved.

3.4.3 Run Statistics

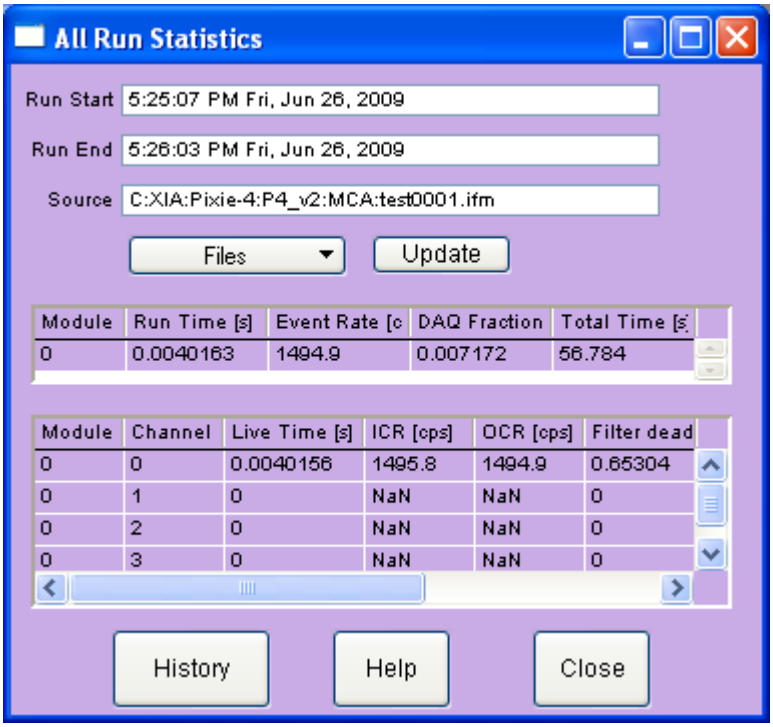

All Run Statistics
Run Start 5:25:07 PM Fri, Jun 26, 2009
Run End 5:26:03 PM Fri, Jun 26, 2009
Source C:XIA:Pixie-4:P4_v2:MCA:test0001.ifm
Files
Update
Module Run Time [s] Event Rate [c DAQ Fraction Total Time [s
0 0.0040163 1494.9 0.007172 56.784
Module Channel Live Time [s] ICR [cps] OCR [cps] Filter dead
0 0 0.0040156 1495.8 1494.9 0.65304
0 1 0 NaN NaN 0
0 2 0 NaN NaN 0
0 3 0 NaN NaN 0
History
Help
Close

Figure 3.8: The Run Statistics panel.

The Run Statistics panel shows the live times and count rates measured by the Pixie-4. The numbers can be updated by clicking the *Update* button and read from or save to *Files*. For a detailed description of the definition of these values, see section 6.6.

3.4.4 File Series

See section 3.6 for a more detailed description

3.5 Optimizing Parameters

Optimization of the Pixie-4's run parameters for best resolution depends on the individual systems and usually requires some degree of experimentation. The Pixie Viewer includes several diagnostic tools and settings options to assist the user, as described below.

3.5.1 Noise

For a quick analysis of the electronic noise in the system, you can view a Fourier transform of the incoming signal by selecting Oscilloscope → *FFT*. The graph shows the FFT of the untriggered input sigal of the Oscilloscope. By adjusting the *dT* control in the Oscilloscope and clicking the *Refresh* button, you can investigate different frequency ranges. For best results, remove any source from the detector and only regard traces without actual events. If you find sharp lines in the 10 kHz to 1 MHz region you may need to find the cause for this and remove it. If you click on the *Apply Filter* button, you can see the effect of the energy filter simulated on the noise spectrum.

© XIA 2012. All rights reserved.

3.5.2 Energy Filter Parameters

The main parameter to optimize energy resolution is the energy filter rise time. Generally, longer rise times result in better resolution, but reduce the throughput. Optimization should begin with scanning the rise time through the available range. Try 2µs, 4µs, 8µs, 11.2µs, take a run of 60s or so for each and note changes in energy resolution. Then fine tune the rise time.

The flat top usually needs only small adjustments. For a typical coaxial Ge-detector we suggest to use a flat top of 1.2µs. For a small detector (20% efficiency) a flat top of 0.8µs is a good choice. For larger detectors flat tops of 1.2µs and 1.6µs will be more appropriate. In general the flat top needs to be wide enough to accommodate the longest typical signal rise time from the detector. It then needs to be wider by one filter clock cycle than that minimum, but at least 3 filter clock cycles. Note that a filter clock cycle ranges from 0.026 to 0.853µs, depending on the filter range, so that it is not possible to have a very short flat top together with a very long filter rise time.

The Pixie Viewer provides a tool which automatically scans all possible combinations of energy filter rise time and flat top and finds the combination that gives the best energy resolution. This tool can be accessed by clicking the *Optimize* button on the *Settings* tab. Please refer to the Online Help documentation for more details. A second option is to create a file series where the energy filter parameters are modified for each file in the series. See section 3.6 for more details.

3.5.3 Threshold and Trigger Filter Parameters

In general, the trigger threshold should be set as low as possible for best resolution. If too low, the input count rate will go up dramatically and "noise peaks" will appear at the low energy end of the spectrum. If the threshold is too high, especially at high count rates, low energy events below the threshold can pass the pile-up inspector and pile up with larger events. This increases the measured energy and thus leads to exponential tails on the (ideally Gaussian) peaks in the spectrum. Ideally, the threshold should be set such that the noise peaks just disappear.

The settings of the trigger filter have only minor effect on the resolution. However, changing the trigger conditions might have some effect on certain undesirable peak shapes. A longer trigger rise time allows the threshold to be lowered more, since the noise is averaged over longer periods. This can help to remove tails on the peaks. A long trigger flat top will help to trigger better on slow rising pulses and thus result in a sharper cut off at the threshold in the spectrum.

3.5.4 Decay Time

The preamplifier decay time τ is used to correct the energy of a pulse sitting on the falling slope of a previous pulse. The calculations assume a simple exponential decay with one decay constant. A precise value of τ is especially important at high count rates where pulses overlap more frequently. If τ is off the optimum, peaks in the spectrum will broaden, and if τ is very wrong, the spectrum will be significantly blurred.

The first and usually sufficiently precise estimate of τ can be obtained from the *Auto Find* routine in the *Energy* tab of the PARAMETER SETUP panel. Measure the decay time several times and settle on the average value.

© XIA 2012. All rights reserved.

Fine tuning of τ can be achieved by exploring small variations around the fit value (±2-3%). This is best done at high count rates, as the effect on the resolution is more pronounced. The value of τ found through this way is also valid for low count rates. Manually enter τ , take a short run, and note the value of τ that gives the best resolution.

Pixie users can also use the fit routines in the OSCILLOSCOPE to manually find the decay time through exponentially fitting the untriggered input signals. Another tool is the *Optimize* routine in the *Energy* tab of the PARAMETER SETUP panel. Similar to the routine for finding the optimal energy filter times, this routine can be used to automatically scan a range of decay times and find the optimal one. Please refer to the Online Help documentation for more details. A further option is to create a file series where τ is modified for each file in the series. See section 3.6 for more details.

3.6 File Series

3.6.1 File Series to break up long data acquisition runs

When taking long data acquisitions, it may be beneficial to break up the run into smaller sub runs. This helps to save data in case of power failure or system crashes, since only the most recent sub run is lost. Also list mode files tend to get large and unwieldy for analysis in longer runs.

The Pixie Viewer thus has a method to create a series of files at specified intervals. In the DATA RECORD OPTIONS panel, opened with the *Record* button in the *Run Control* tab of the PARAMETER SETUP panel, there is a checkbox named *New files every*, followed by a control field to enter a spill (or time) interval N. If checked and a run is started, every N spills (or, in MCA runs, every N seconds) the data file is closed, spectra, settings and statistics are saved, and then a new run is started. This is equivalent to manually clicking first the Stop Run button and then the Start Run button. It is recommended to enable the automatic increment and auto-store options as shown in Figure 3.9 as well.

© XIA 2012. All rights reserved.

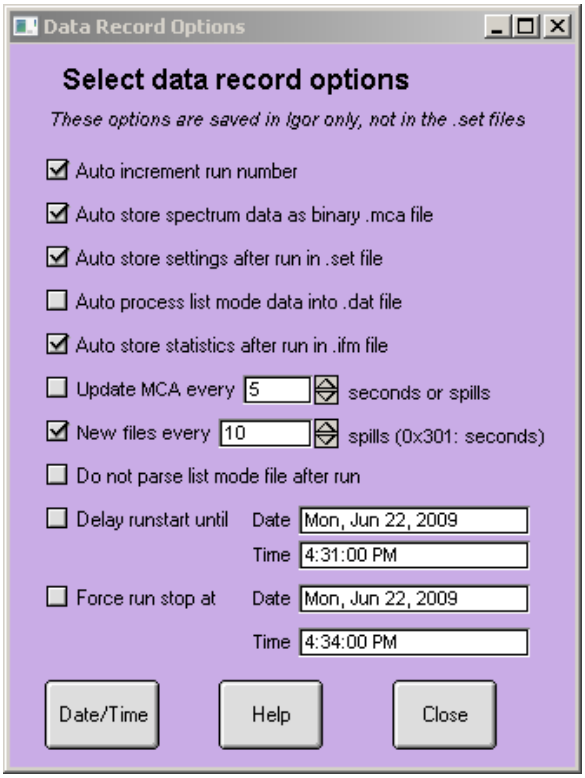

Data Record Options
Select data record options
These options are saved in Igor only, not in the .set files
Auto increment run number
Auto store spectrum data as binary .mca file
Auto store settings after run in .set file
Auto process list mode data into .dat file
Auto store statistics after run in .ifm file
Update MCA every 5 seconds or spills
New files every 10 spills (0x301: seconds)
Do not parse list mode file after run
Delay runstart until Date Mon, Jun 22, 2009
Time 4:31:00 PM
Force run stop at Date Mon, Jun 22, 2009
Time 4:34:00 PM
Date/Time
Help
Close

Figure 3.9: The DATA RECORD OPTIONS panel with checkboxes set to acquire a series of files.

3.6.2 File Series to scan filter parameters

With some modifications, the mechanism to create file series described in section 3.6.1 can also be used to scan through a range of energy filter or decay time settings. This is equivalent to starting an MCA run with initial settings, stopping the run, incrementing the energy filter rise time, restarting the run, and so on. The file series will thus contain spectra for a whole range of settings, which can be analyzed manually or with the routine described in section 3.6.3.

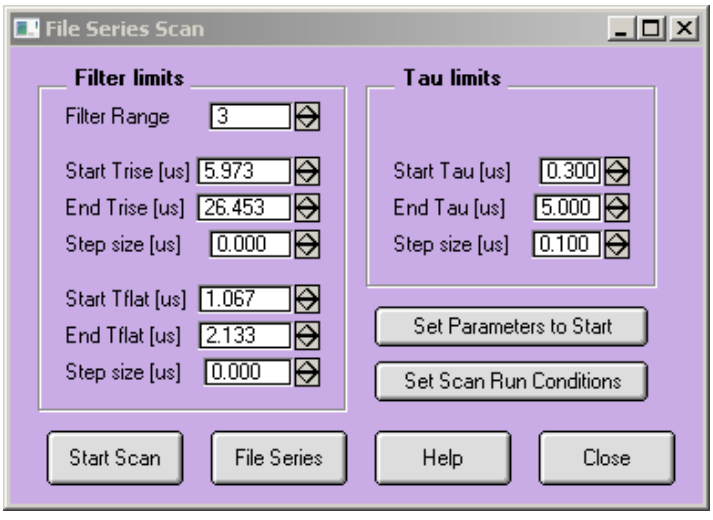


Figure 3.10: The FILE SERIES SCAN panel to acquire a series of files in which energy filter parameters and Tau are varied within user defined limits.

© XIA 2012. All rights reserved.

To set up such a parameter scan, open the FILE SERIES SCAN panel (PARAMETER SETUP -> *Energy* tab -> *Scan Settings*) shown in Figure 3.10. A control field named *Filter Range* is repeated from the *Energy* tab. In three groups of controls, you can set the start, end, and step size for varying the energy filter *rise time*, the energy filter *flat top*, and *Tau*. If the *step size* is zero, that parameter will not be varied.

Two buttons assist in setting up the initial conditions: *Set Parameters to Start* sets the current values of the energy filter and Tau to the start value defined in the File Series Scan panel. If you omit to click this button, the file series will begin with the current value; this is useful to resume a file series. *Set Scan Run Conditions* will set the checkboxes in the DATA RECORD OPTIONS panel to the values required for the scan, and set the run time to the total time required (interval N in the DATA RECORD OPTIONS panel times the number of settings).

At the bottom of the panel, the button *Start Scan* starts the file series. This is a different button from the standard *Start Run* button, because it is starting a run which is modifying parameters. All the updates during a run work the same as in a standard run, though; and the run can be stopped with the standard *Stop Run* button. When the run is complete, click on the *File Series* button to open the panel described in section 3.6.3

3.6.3 File Series Analysis

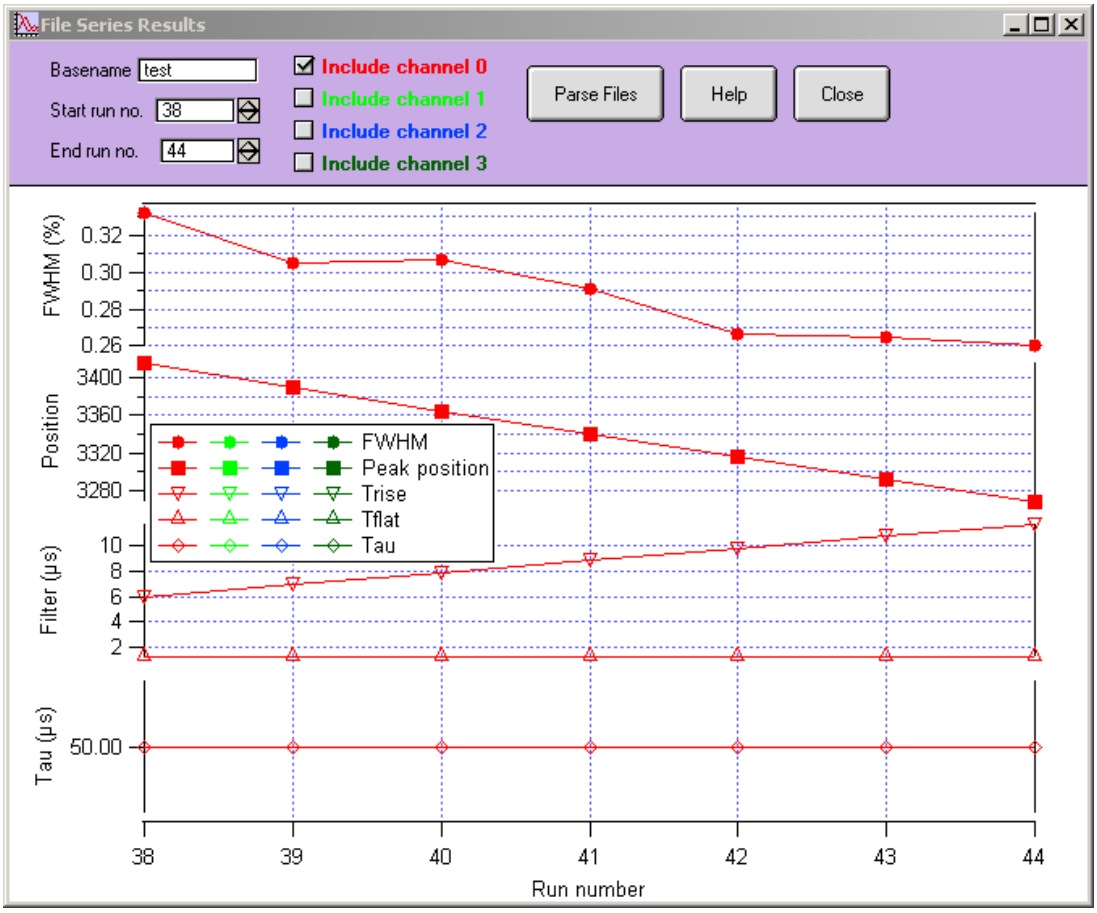


Figure 3.11: The FILE SERIES RESULTS plot to analyze a series of files from a parameter scan.

© XIA 2012. All rights reserved.

To analyze a series of .mca files, you can use the FILE SERIES RESULTS panel. Enter the base name and the start and end run numbers of the series, then click *Parse Files*. Start and end are inclusive, i.e. for start = 1 and end =13, the parsing covers files base0001-base0013. An .ifm file is required to read the values for Tau and the filter settings. The parsing routine reads the spectra and fits peaks with the options set in the MCA SPECTRUM. Thus make sure the fit range is set appropriately. In the plot, the peak position and resolution is plotted as a function of run number, together with the filter settings and tau for each channel selected.

© XIA 2012. All rights reserved.

4 Data Runs and Data Structures

4.1 Run Types

There are two major run types: MCA runs and List mode runs. MCA runs only collect spectra, List mode runs acquire data on an event-by event basis, but also collect spectra. List mode runs come in several variants (see below).

The output data are available in three different memory blocks. The multichannel analyzer (MCA) block resides in memory external to the DSP. There is a local I/O data buffer for list mode data located in the DSP, consisting of 8192 16-bit words, and an extended I/O data buffer for list mode runs in the external memory, holding up to 32 local buffers.

4.1.1 MCA Runs

If all you want to do is to collect spectra, you should start an MCA run. For each event, this type of run collects the data necessary to calculate pulse heights (energies) only. The energy values are used to increment the MCA spectrum. The run continues until the host computer stops data acquisition, either by reaching the run time set in the Pixie Viewer, or by a manual stop from the user (the module does not stop by itself). Run statistics, such as live time, run time, and count rates are kept in the Pixie module.

4.1.2 List Mode Runs

If, on the other hand, you want to operate the Pixie in multi-parametric or list mode and collect data on an event-by-event basis, including energies, time stamps, pulse shape analysis values, and wave forms, you should start a list mode run. In list mode, you can still request histogramming of energies, e.g. for monitoring purposes. In the current standard software, one pulse shape analysis value is a constant fraction trigger time calculated by the DSP, the other is reserved for user-written event processing routines. Other routines exist e.g. to calculate rise times and/or to characterize pulses from phoswich detectors.

List mode runs halt data acquisition either when the local I/O data buffer is full, or when a preset number of events are reached. The maximum number of events (MAXEVENTS) is calculated by the Pixie Viewer when selecting a run type, shown in the control *Events/Buffer*, and downloaded to the module as the preset number before starting a run. This default value for MAXEVENTS is the maximum "safe" number of events. That is, given the maximum length of an event (all "good" channels contributing), in any case at least MAXEVENTS events will fit in the output buffer. MAXEVENTS can be decreased by the user if desired. MAXEVENTS can be set to zero, to disable the halting at a preset number. This makes the acquisition more efficient: if MAXEVENTS takes into account 4 “good” channels per event, but if in the acquisition only few multi-channel events occur, the buffer will only filled up to about 1/4 when MAXEVENTS is reached. Setting MAXEVENTS to zero will always fill the buffer as much as possible.

© XIA 2012. All rights reserved.

4.1.2.1 Compressed Data Formats

The output data of list mode runs can be reduced by using one of the compressed formats described below. The key difference is that as less data is recorded for each event, there is room for more events in the I/O data buffer of the Pixie-4 module and less time is spent per event to read out data to the host computer. For example, if you need individual energies and time stamps, but no waveforms, select run type "0x103 energy and time only" instead of "0x100 general purpose" in the Pixie Viewer. In run type 0x103, raw data from the energy filters and waveforms are temporarily stored in an intermediate buffer, and only results are written to the output buffer.

In compressed list mode runs, the following points are to consider:

- When using a runtype that computes results of pulse shape analysis (PSA) computations, make sure the total combined trace length from all four channels is less than 52µs because the intermediate buffer used to temporarily store the trace data is limited to ~4K samples.
- If no PSA is required, reduce the trace length to zero to avoid unnecessary data transfer time.
- When the intermediate buffer is filled with events not yet processed for output data, new events are rejected. Whenever a new event occurs, the DSP first checks if there is enough room left in the intermediate buffer, then transfers the data from the FPGAs into its intermediate buffer or rejects it. Consequently, if the combined trace length is more than 26µs, only one event at a time can be stored. This means that the effective dead time for an event is increased by the processing time. If the combined trace length is such that N >= 2 events fit into the intermediate buffer, the processing time does not add to the dead time as long as the *average* event rate is smaller than the processing rate and no bursts of more than N events occur.

4.1.2.2 Data Readout Options

List mode runs halt data acquisition either when the local I/O data buffer is full, or when a preset number of events are reached. The data has then to be read out by the host PC. (Runs can be resumed for longer data acquisitions as described in 4.1.2.3). There are three options for the data readout mode, with different consequences for the readout dead time. (Modes 2 and 3 are only available for module revisions C and D.)

a) The default readout option available for all module revisions is to simply fill and read the local I/O buffer. However, data readout from the local I/O buffer is relatively slow, and since acquisition is halted during the readout, the readout dead time is relatively large (~30ms per module and buffer).

b) A more efficient readout mode is to transfer the data is to the external memory when the local buffer is full, and resume the run right away. This is repeated 32 times until the external memory is full, and only then the run is halted and data is read out by the host PC in a fast block read (~30ms per 32 buffers and module, about 550µs between buffers).

c) A third readout mode is to transfer the data is to the external memory when the local buffer is full, and resume the run as in mode 2, but after 16 times a flag is raised to the host PC to read out the external memory while new data is stored in the other half of the

© XIA 2012. All rights reserved.

external memory. This allows (almost) uninterrupted data acquisition. Readout dead time between buffers is about 550 µs, and readout time for 16 buffers is again ~30ms. Note that at high cont rates and/or for uncompressed data runs with long waveforms, 16 buffers may fill up faster than 30ms. Readout mode 3 is only efficient as long as the time to half fill the external memory is longer than the readout time.

4.1.2.3 Multiple Spills

Runs can be "resumed" by the host after the data is read out. In a resumed run, run statistics are not cleared at the beginning of the run, i.e. it is possible to combine several buffer readouts ("spills") into one extended run, appended to the same file. In the Pixie Viewer this is done automatically when requesting several spills.

Note that for runs with several modules, the buffer ordering in the data depends on the readout option. In mode 1, the data file begins with the first buffer readout of module 0, followed by first buffers of module 1, module 2, … module N, then the second buffers of modules 0 to N, and so forth. In readout mode 2, list mode runs are repeated 32 times before readout. Therefore the data file will begin with the first **32** buffer readouts of module 0, followed by the first **32** buffers of module 1, module 2, … module N, then a second **32** buffers of module 0 to N and so forth. In readout mode 3 above, list mode runs are repeated 16 times before readout. Therefore the data file will begin with the first **16** buffer readouts of module 0, followed by the first **16** buffers of module 1, module 2, … module N, then a second **16** buffers of module 0 to N and so forth.

Table 4.1: Summary of run types and data formats.

Run Type	Output data	DSP Variables
List Mode (standard)	Energies, time stamps, 6 PSA values, and wave forms in List mode block. Spectra in MCA block	RUNTASK = 256 MAXEVENTS = <calculate> (CHANHEADLEN = 9)
List Mode Compression 1	Energies, time stamps, and 6 PSA values in List mode block. Spectra in MCA block	RUNTASK = 257 MAXEVENTS = <calculate> (CHANHEADLEN = 9)
List Mode Compression 2	Energies, time stamps, and 2 PSA values in List mode block. Spectra in MCA block	RUNTASK = 258 MAXEVENTS = <calculate> (CHANHEADLEN = 4)
List Mode Compression 3	Energies and time stamps in List mode block. Spectra in MCA block	RUNTASK = 259 MAXEVENTS = <calculate> (CHANHEADLEN = 2)
MCA Mode	Spectra in MCA block	RUNTASK = 769 MAXEVENTS=0

4.1.3 Fast List Mode Runs

The legacy "fast list mode runs" are no longer supported.

4.2 Output Data Structures

4.2.1 MCA Histogram Data

The MCA block is fixed to 32K words (32-bit deep) per channel, i.e. total 128K words. The MCA block resides in the external memory which can be read out via the PCI data bus at rates

© XIA 2012. All rights reserved.

over 100Mbytes/s. If spectra of less than 32K length are requested, only part of the 32K will be filled with data. This data can be read even when a run is in progress, to get a spectrum update.

In clover mode, spectra for each channel are 16K long and compressed into the first 64K of the external memory. An additional 16K addback spectrum containing the sum of energies in events with multiple hits is accumulated in the second 64K of the external memory.

In 2D mode, spectra for each channel are 16K long and compressed into the first 64K of the external memory. The remaining 64K of the external memory contains a 256 x 256 bin two dimensional spectrum (custom code required)

4.2.2 List Mode Data

The list mode data in external memory consists of 32 local I/O data buffers. The local I/O data buffer can be written by the DSP in a number of formats. User code should access the three variables BUFHEADLEN, EVENTHEADLEN, and CHANHEADLEN in the configuration file of a particular run to navigate through the data set. It should only be read when the run has ended.

The 32 buffers in external memory follow immediately one after the other. The data organization of one I/O buffer is as follows. The buffer content always starts with a buffer header of length BUFHEADLEN. Currently, BUFHEADLEN is six, and the six words are:

Table 4.2: Buffer header data format.

Word #	Variable	Description
0	BUF_NDATA	Number of words in this buffer
1	BUF_MODNUM	Module number
2	BUF_FORMAT	Format descriptor = RunTask + 0x20T0 T: bits4-7 indicate channel 0-3 acquired waveforms in 4x trace mode.
3	BUF_TIMEHI	Run start time, high word
4	BUF_TIMEMI	Run start time, middle word
5	BUF_TIMELO	Run start time, low word

Following the buffer header, the events are stored in sequential order. Each event starts out with an event header of length EVENTHEADLEN. Currently, EVENTHEADLEN=3, and the three words are:

Table 4.3: Event header data format.

Word #	Variable	Description
0	EVT_PATTERN	Hit pattern. Bit [15..0] = [gate pattern | hit pattern | status | read pattern]
1	EVT_TIMEHI	Event time, high word
2	EVT_TIMELO	Event time, low word

The hit pattern is a bit mask, which tells which channels were read out plus some additional status information, as listed in table 4.4. After the event header follows the channel information as indicated by the hit pattern, in order of increasing channel numbers. For example, if bits[3:0] = 1001, the event header is followed by data from channel 0, then channel 3.

© XIA 2012. All rights reserved.

Table 4.4: Hit pattern bit description.

Bit #	Description
0..3	If set, indicates that data for channel 0..3 have been recorded
4..7	4: Logic level of FRONT panel input 5: Result of LOCAL coincidence test 6: Logic level of backplane STATUS line, 7: Logic level of backplane TOKEN line (= result of global coincidence test), see section 7
8..11	If set, indicates that channel 0..3 has been hit in this event (i.e. if zero, energy reported is invalid or only an estimate)
12..15	Logic level of the GATE input of channel 0..3 (for Rev. D modules only).

The data for each channel are organized into a channel header of length CHANHEADLEN, which may be followed by waveform data. CHANHEADLEN depends on the run type and on the method of data buffering, i.e. if raw data is directed to the intermediate Level-1 buffer or directly to the linear buffer. Offline analysis programs should therefore check the value of RUNTASK, which is reported in the buffer header. All currently supported data formats are defined below.

1. For List Mode, either standard or compression 1, (RUNTASK = 256, 257), CHANHEADLEN=9, and the nine words are

   Table 4.5: Channel header, possibly followed by waveform data.

Word #	Variable	Description
0	CHAN_NDATA	Number of words for this channel
1	CHAN_TRIGTIME	Fast trigger time
2	CHAN_ENERGY	Energy
3	CHAN_XIAPSA	XIA PSA value
4	CHAN_USERPSA	User PSA value
5	Unused	N/A
6	Unused	N/A
7	Unused	N/A
8	CHAN_REALTIMEHI	High word of the real time

   Any waveform data for this channel would then follow this header. An offline analysis program can recognize this by computing
   N_WAVE_DATA = CHAN_NDATA- CHANHEADLEN.
   If N_WAVE_DATA is greater than zero, it indicates the number of waveform data words to follow.
   In the current software version, the XIA PSA value contains the result of the constant fraction trigger time computation (CFD). The format is as follows: the upper 8-bit of the word point to the ADC sample before the CFD, counted from the beginning of the trace. The lower 8 bits give the fraction of an ADC sample time between the sample and the CFD time. For example, if the value is 0x0509, the CFD time is 5 + 9/256 ADC sample steps away from the beginning of the recorded trace.
   The User PSA value contains flags to indicate special events if it is not overwritten by User DSP code. (CHAN_USERPSA bits 3|2|1|0 = Gate | Pileup | Out of range | Veto).

© XIA 2012. All rights reserved.

Normally, events that are piled up etc are rejected from the acquisition and do not appear in the list mode data. However, when the advanced options to allow pileup or out of range are enabled or when a VETO or GATE signal is present but rejection is not enabled, such events are recorded. The data in User_PSA can be used to find such events and treat them in special ways.

2. For compression 2 List Mode (RUNTASK = 258), CHANHEADLEN=4, and the four words are:

Table 4.6: Channel header for compression 2 format.

Word #	Variable	Description
1	CHAN_TRIGTIME	Fast trigger time
2	CHAN_ENERGY	Energy
3	CHAN_XIAPSA	XIA PSA value
4	CHAN_USERPSA	User PSA value

3. For compression 3 List Mode (RUNTASK = 259), CHANHEADLEN=2, and the two words are:

Table 4.7: Channel header for compression 3 format.

Word #	Variable	Description
1	CHAN_TRIGTIME	Fast trigger time
2	CHAN_ENERGY	Energy

4.2.3 Reconstruction of list mode time stamps

As discussed above, in list mode the Pixie-4 records time information in three different locations: The buffer header, the event header and the channel header. Below we describe how to combine these time stamps into full timing information.

In the Pixie-4, there is a 48-bit time counter that is reset to zero at boot time or at a run start with the "synchronize clocks" option selected. It is incremented at the full processor clock rate of 75 MHz; the unit of the LSB is one 13.33ns[5]. Hence, the 48-bit word can span a time interval of 43.44 days before rolling over.

At the beginning of a data acquisition run, the module looks at the counter and records the BufferTime as a 48-bit number, using three 16-bit words. These three words are the time stamps in the buffer header (BUF_TIMEHI, BUF_TIMEMI, BUF_TIMELO). Every time an event is recognized, the DSP records the lower 32-bit of the time counter as two 16-bit words in the event header (EVT_TIMEHI, EVT_TIMELO). Every time a channel triggers at the rising edge of a pulse, the channel records the lowest 16 bit of the time counter, which is reported in the channel header (CHAN_TRIGTIME). As the rising edge occurs a filter time before the event is recognized, the channel time is typically a few microseconds ahead of the event time.

Using C-parlance and enumerating beginning at 0, we would reconstruct the true time of arrival for any event as:

EventTime = EVT_TIMELO;

[5]For best precision, use 1us/75 in the conversion from clock ticks to seconds, 13.33e-9s may lead to rounding errors.

© XIA 2012. All rights reserved.

EventTime += EVT_TIMEHI*pow(2,16);

EventTime += BUF_TIMEHI*pow(2,32);

EventTime *= 1e-6/75s;

This EventTime can be used to match events between modules and to compute time differences between events.

ChannelTime serves as a refinement to compute the time of arrival differences between channels of the same event. For most purposes, it is sufficient (and more convenient) to simply compute a relative time difference, e.g.

$ChannelTimeDiff_{0,1}$ = (CHAN_TRIGTIME $_0$ - $CHAN\_TRIGTIME_1$)*1e-6/75s

However, it has to be kept in mind that the 16-bit word may overflow between being recorded in channel 0 and channel 1. For example, channel 0 may trigger first at CHAN_TRIGTIME = 65500 and channel 1 may trigger later at CHAN_TRIGTIME = 105 and thus $ChannelTimeDiff_{0,1}$ would be 65395 * 1e-6/75s even though the correct time difference would only be 141 * 1e-6/75s. If such large time differences can be known to be impossible from the physics of the experiment (e.g. max. time of flight delay) or from the acquisition setup (e.g. max. coincidence window), such overflows can be handled by simply adding 65536 to the smaller CHAN_TRIGTIME if the time difference is greater than the known maximum (e.g. 63000 * 1e-6/75s).

If an absolute ChannelTime is required, it can be computed as follows:

ChannelTime = CHAN_TRIGTIME;

ChannelTime += EVT_TIMEHI*pow(2,16);

ChannelTime += BUF_TIMEHI*pow(2,32);

ChannelTime *= 1e-6/75s;

For a small fraction of events, the EventTime or the absolute ChannelTime will experience an overflow in a lower word in one header which is not reflected in the next higher word recorded in a different header. These can typically be easily recognized and corrected for.

1) The channel time is recorded before the event time. Hence, the time counter may have seen a 16 bit roll over (change from high to low values of the low word) between the recording of the channel time stamp and the recording of the event time stamps. For example, a channel may trigger at CHAN_TRIGTIME = 65100 and the event may be recognized after that at (EVT_TIMEHI, EVT_TIMELO) = (5467, 123). The correct absolute ChannelTime would be reconstructed from <BUF_TIMEHI>, 5466, and 65100, i.e. using EVT_TIMEHI-1 instead of EVT_TIMEHI as the middle word. So if ChannelTime comes out larger (later) than the EventTime, it needs to be reduced by 65536* 1e-6/75s
2) The buffer start time is recorded before the event time. Hence, the time counter may have seen a 32 bit roll over (change from high to low values of the middle word) between the recording of the buffer time stamps and the recording of the event time stamps. For example, the run may have started at (BUF_TIMEHI, BUF_TIMEMI, BUF_TIMELO)

© XIA 2012. All rights reserved.

= (2,65510,34524) and the event may be recognized after that at (EVT_TIMEHI, EVT_TIMELO) = (7, 7564). The correct EventTime would be reconstructed from 3, 7, and 7564, i.e. using BUF_TIMEHI+1 instead of BUF_TIMEHI as the high word. So if EventTime comes out smaller (earlier) than the BufferTime or the EventTime of a previous event in the same buffer, it needs to be increased by 2^32* 1e-6/75s.

3) If count rates are extremely low, in the order of one count per minute or less, the time counter may see multiple 32-bit rollovers events. For example the run may have started at (3, 153, 17564) and the first event may occur at (8, 56, 456). With only the lower two words recorded in the event header, the rule described in 2) above (use BUF_TIMEHI+1) would lead to incorrectly using "4" as the highest 16 bit word in the computation of EventTime. In this case, CHAN_REALTIMEHI should be used instead of BUF_TIMEHI. It contains the highest 16 bit word at the time the event is processed by the DSP, in this example the correct[6] value "8". However, this word is only available in run types 0x100 and 0x101.

[6]Since CHAN_REALTIMEHI is recorded several microseconds after the event time, there is a ~1 in 1e6 chance of it being off by 1

© XIA 2012. All rights reserved.

5 Hardware Description

The Pixie-4 is a 4-channel unit designed for gamma-ray spectroscopy and waveform capturing. It incorporates four functional building blocks, which we describe below. This section concentrates on the functionality aspect. Technical specification can be found in section 1.2. Figure 5.1 shows the functional block diagram of the Pixie-4.

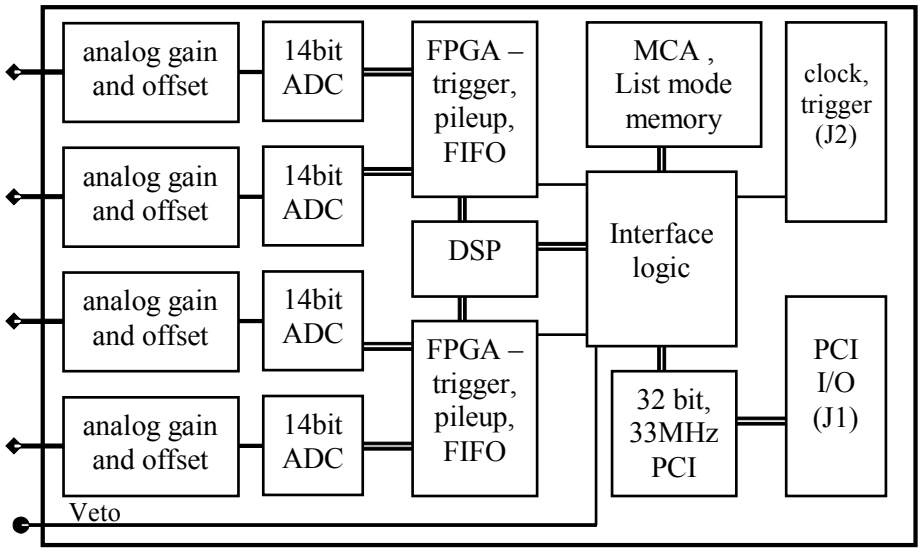


Figure 5.1: Functional block diagram of the Pixie-4 front-end data acquisition and signal processing card.

5.1 Analog Signal Conditioning

Each analog input has its own signal conditioning unit. The task of this circuitry is to adapt the incoming signals to the input voltage range of the ADC, which spans 2V. Input signals are adjusted for offsets, and there is a computer-controlled gain stage of switched relays. This helps to bring the signals into the ADC's voltage range and set the dynamic range of the channel. A fine tuning of the gain is achieved by multiplying the calculated energy values with digital gain factors in the digital signal processor (DSP).

The ADC is not a peak sensing ADC, but acts as a waveform digitizer. In order to avoid aliasing, we remove the high frequency components from the incoming signal prior to feeding it into the ADC. The anti-aliasing filter, an active Sallen-Key filter, cuts off sharply at the Nyquist frequency, namely half the ADC sampling frequency.

Though the Pixie-4 can work with many different signal forms, best performance is to be expected when sending the output from a charge integrating preamplifier directly to the Pixie-4 without any further shaping.

© XIA 2012. All rights reserved.

5.2 Real-time Processing Units

The real time processing units (RTPUs), one per two channels, consist of a field programmable gate array (FPGA) which also incorporates a FIFO memory for each channel. The data stream from the ADCs is sent to these units at the full ADC sampling rate. Using a pipelined architecture, the signals are processed at this high rate, without the help of the on-board DSP.

Note that the use of one RTPU for two channels allows sampling the incoming signal at twice the regular ADC clock frequency. If both channels are fed the same input signal, the RTPU can give the second ADC a clock with a 180 degree phase shift, thus sampling the signal twice in one clock cycle. Special firmware is required to combine the two input streams into one; contact XIA for details.

The RTPUs apply digital filtering to perform essentially the same action as a shaping amplifier. The important difference is in the type of filter used. In a digital application it is easy to implement finite impulse response filters, and we use a trapezoidal filter. The flat top will typically cover the rise time of the incoming signal and makes the pulse height measurement less sensitive to variations of the signal shape.

Secondly, the RTPUs contain a pileup inspector. This logic ensures that if a second pulse is detected too soon after the first, so that it would corrupt the first pulse height measurement, both pulses are rejected as piled up. The pileup inspector is, however, not very effective in detecting pulse pileup on the rising edge of the first pulse, i.e. in general pulses must be separated by their rise time to be effectively recognized as different pulses. Therefore, for high count rate applications, the pulse rise times should be as short as possible, to minimize the occurrence of pileup peaks in the resulting spectra.

If a pulse was detected and passed the pileup inspector, a trigger may be issued. That trigger would notify the DSP that there are raw data available now. If a trigger was issued the data remain latched until the RTPU has been serviced by the DSP.

The third component of the RTPU is a FIFO memory, which is controlled by the pile up inspector logic. The FIFO memory is continuously being filled with waveform data from the ADC, only stopped to avoid overwriting of data for valid events. On a trigger the read pointer is positioned such that it points to the beginning of the pulse that caused the trigger. When the DSP collects event data, it can read any fraction of the stored waveform, up to the full length of the FIFO.

5.3 Digital Signal Processor (DSP)

The DSP controls the operation of the Pixie-4, reads raw data from the RTPUs, reconstructs true pulse heights, applies time stamps, prepares data for output to the host computer, and increments spectra in the on-board memory.

The host computer communicates with the DSP, via the PCI interface, using a direct memory access (DMA) channel. Reading and writing data to DSP memory does not interrupt its operation, and can occur even while a measurement is underway.

© XIA 2012. All rights reserved.

The host sets variables in the DSP memory and if necessary calls DSP functions to apply them to the hardware. Through this mechanism all gain and offset DACs are set and the filter settings are applied to the RTPUs.

The RTPUs process their data without support from the DSP, once they have been set up. When any one or more of them generate a trigger, an interrupt request is sent to the DSP. It responds with reading the required raw data from the RTPUs and storing those in an intermediate buffer. It then returns from the interrupt routine without processing the data to minimize the DSP induced dead time. The event processing routine works from the data in the buffer to generate the requested output data. There are different implementations of the intermediate buffer for the different run types. In standard list mode runs, intermediate and I/O buffer are the same, to avoid moving long waveforms inside the DSP. In other run types, the intermediate buffer is circular, i.e. old data is overwritten once it has been processed. If the circular buffer fills up before the data can be processed, no further raw data is read from the RTPUs.

In this scheme, the greatest processing power is located in the RTPUs. Implemented in FPGAs each of them processes the incoming waveforms from its associated ADC in real time and produces, for each valid a event, a small set of distilled data from which pulse heights and arrival times can be reconstructed. The computational load for the DSP is much reduced, as it has to react only on an event-by-event basis and has to work with only a small set of numbers for each event.

5.4 PCI Interface

The PCI interface through which the host communicates with the Pixie-4 is implemented in a PCI slave IC together with an FPGA. The configuration of this PCI IC is stored in a PROM, which is placed in the only DIP-8 IC-socket on the Pixie-4 board. The interface conforms to the commercial PCI standard. It moves 32-bit data words at a time.

The interface does not issue interrupt requests to the host computer. Instead, for example to determine when data is ready for readout, the host has to poll a Control and Status Register (CSR) in the interface logic, also called communication FPGA.

The communication FPGA links the PCI slave with the DSP and the on-board memory. The host can read out the memory without interrupting the operation of the DSP. This allows updates of the MCA spectrum while a run is in progress. The communication FPGA also distributes triggers and coincidence signals to other modules using the PXI backplane connections.

© XIA 2012. All rights reserved.

6 Theory of Operation

6.1 Digital Filters for γ-ray Detectors

Energy dispersive detectors, which include such solid state detectors as Si(Li), HPGe, HgI_2, CdTe and CZT detectors, are generally operated with charge sensitive preamplifiers as shown in Figure 6.1 (a). Here the detector D is biased by voltage source V and connected to the input of preamplifier A which has feedback capacitor C_f and feedback resistor R_f.

The output of the preamplifier following the absorption of an γ-ray of energy E_x in detector D is shown in Figure 6.1 (b) as a step of amplitude V_x (on a longer time scale, the step will decay exponentially back to the baseline, see section 6.3). When the γ-ray is absorbed in the detector material it releases an electric charge $Q_x = E_x/\varepsilon$, where ε is a material constant. Q_x is integrated onto C_f, to produce the voltage $V_x = Q_x/C_f = E_x/(\varepsilon C_f)$. Measuring the energy E_x of the γ-ray therefore requires a measurement of the voltage step V_x in the presence of the amplifier noise σ, as indicated in Figure 6.1 (b).

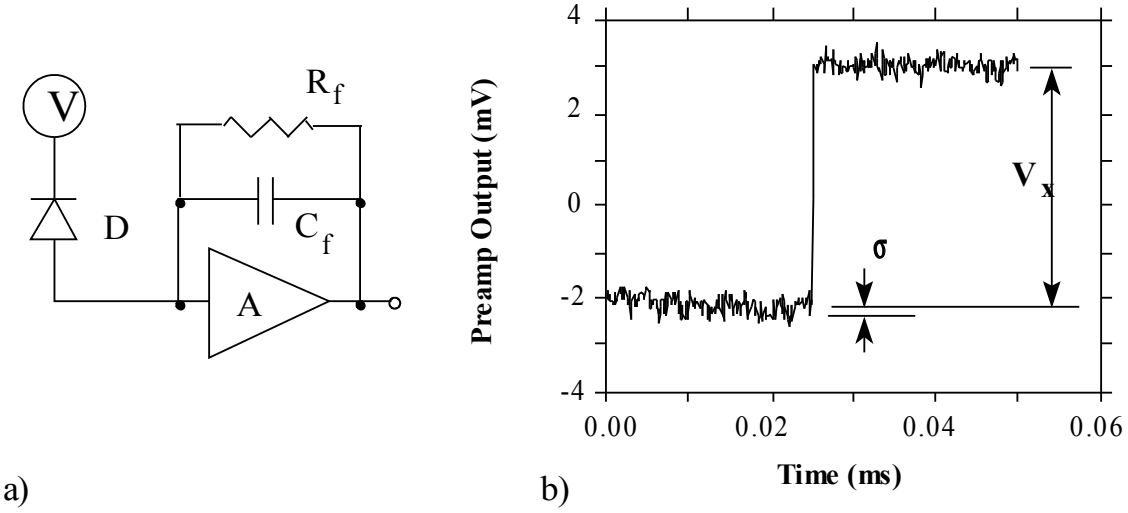


Figure 6.1: (a) Charge sensitive preamplifier with RC feedback; (b) Output on absorption of an γ-ray.

Reducing noise in an electrical measurement is accomplished by filtering. Traditional analog filters use combinations of a differentiation stage and multiple integration stages to convert the preamp output steps, such as shown in Figure 6.1 (b), into either triangular or semi-Gaussian pulses whose amplitudes (with respect to their baselines) are then proportional to V_x and thus to the γ-ray's energy.

Digital filtering proceeds from a slightly different perspective. Here the signal has been digitized and is no longer continuous. Instead it is a string of discrete values as shown in Figure 6.2. Figure 6.2 is actually just a subset of Figure 6.1 (b), in which the signal was digitized by a Tektronix 544 TDS digital oscilloscope at 10 MSPS (mega samples per second). Given this data set, and some

© XIA 2012. All rights reserved.

kind of arithmetic processor, the obvious approach to determining V_x is to take some sort of average over the points before the step and subtract it from the value of the average over the points after the step. That is, as shown in Figure 6.2, averages are computed over the two regions marked "Length" (the "Gap" region is omitted because the signal is changing rapidly here), and their difference taken as a measure of V_x. Thus the value V_x may be found from the equation:

$$V_{x,k} = -\sum_{i(before)} W_i V_i + \sum_{i(after)} W_i V_i \tag{6.1}$$

where the values of the weighting constants W_i determine the type of average being computed. The sums of the values of the two sets of weights must be individually normalized.

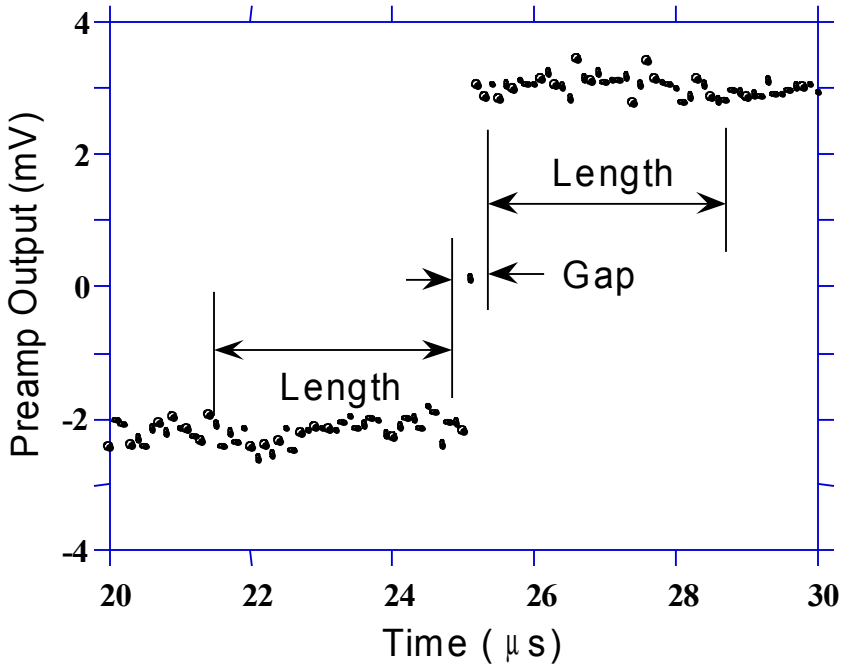


Figure 6.2: Digitized version of the data of Figure 6.1 (b) in the step region.

The primary differences between different digital signal processors lie in two areas: what set of weights $\{W_i\}$ is used and how the regions are selected for the computation of Eqn. 6.1. Thus, for example, when larger weighting values are used for the region close to the step while smaller values are used for the data away from the step, Eqn. 6.1 produces "cusp-like" filters. When the weighting values are constant, one obtains triangular (if the gap is zero) or trapezoidal filters. The concept behind cusp-like filters is that, since the points nearest the step carry the most information about its height, they should be most strongly weighted in the averaging process. How one chooses the filter lengths results in time variant (the lengths vary from pulse to pulse) or time invariant (the lengths are the same for all pulses) filters. Traditional analog filters are time invariant. The concept behind time variant filters is that, since the γ-rays arrive randomly

© XIA 2012. All rights reserved.

and the lengths between them vary accordingly, one can make maximum use of the available information by setting the length to the interpulse spacing.

In principal, the very best filtering is accomplished by using cusp-like weights and time variant filter length selection. There are serious costs associated with this approach however, both in terms of computational power required to evaluate the sums in real time and in the complexity of the electronics required to generate (usually from stored coefficients) normalized $\{W_i\}$ sets on a pulse by pulse basis.

The Pixie-4 takes a different approach because it was optimized for high speed operation. It implements a fixed length filter with all W_i values equal to unity and in fact computes this sum afresh for each new signal value k. Thus the equation implemented is:

$$LV_{x,k} = -\sum_{i=k-2L-G+1}^{k-L-G} V_i + \sum_{i=k-L+1}^{k} V_i \tag{6.2}$$

where the filter length is L and the gap is G. The factor L multiplying $V_{x,k}$ arises because the sum of the weights here is not normalized. Accommodating this factor is trivial.

While this relationship is very simple, it is still very effective. In the first place, this is the digital equivalent of triangular (or trapezoidal if $G \neq 0$) filtering which is the analog industry's standard for high rate processing. In the second place, one can show theoretically that if the noise in the signal is white (i.e. Gaussian distributed) above and below the step, which is typically the case for the short shaping times used for high signal rate processing, then the average in Eqn. 6.2 actually gives the best estimate of V_x in the least squares sense. This, of course, is why triangular filtering has been preferred at high rates. Triangular filtering with time variant filter lengths can, in principle, achieve both somewhat superior resolution and higher throughputs but comes at the cost of a significantly more complex circuit and a rate dependent resolution, which is unacceptable for many types of precise analysis. In practice, XIA's design has been found to duplicate the energy resolution of the best analog shapers while approximately doubling their throughput, providing experimental confirmation of the validity of the approach.

6.2 Trapezoidal Filtering in the Pixie-4

From this point onward, we will only consider trapezoidal filtering as it is implemented in the Pixie-4 according to Eqn. 6.2. The result of applying such a filter with Length L=1μs and Gap G=0.4μs to a γ-ray event is shown in Figure 6.3. The filter output is clearly trapezoidal in shape and has a rise time equal to L, a flattop equal to G, and a symmetrical fall time equal to L. The basewidth, which is a first-order measure of the filter's noise reduction properties, is thus 2L+G.

This raises several important points in comparing the noise performance of the Pixie-4 to analog filtering amplifiers. First, semi-Gaussian filters are usually specified by a *shaping time*. Their rise time is typically twice this and their pulses are not symmetric so that the basewidth is about 5.6 times the shaping time or 2.8 times their rise time. Thus a semi-Gaussian filter typically has a slightly better energy resolution than a triangular filter of the same rise time because it has a longer filtering time. This is typically accommodated in amplifiers offering both triangular and semi-Gaussian filtering by stretching the triangular rise time a bit, so that the *true* triangular rise

© XIA 2012. All rights reserved.

time is typically 1.2 times the selected semi-Gaussian rise time. This also leads to an apparent advantage for the analog system when its energy resolution is compared to a digital system with the same nominal rise time.

One important characteristic of a digitally shaped trapezoidal pulse is its extremely sharp termination on completion of the basewidth 2L+G. This may be compared to analog filtered pulses whose tails may persist up to 40% of the rise time, a phenomenon due to the finite bandwidth of the analog filter. As we shall see below, this sharp termination gives the digital filter a definite rate advantage in pileup free throughput.

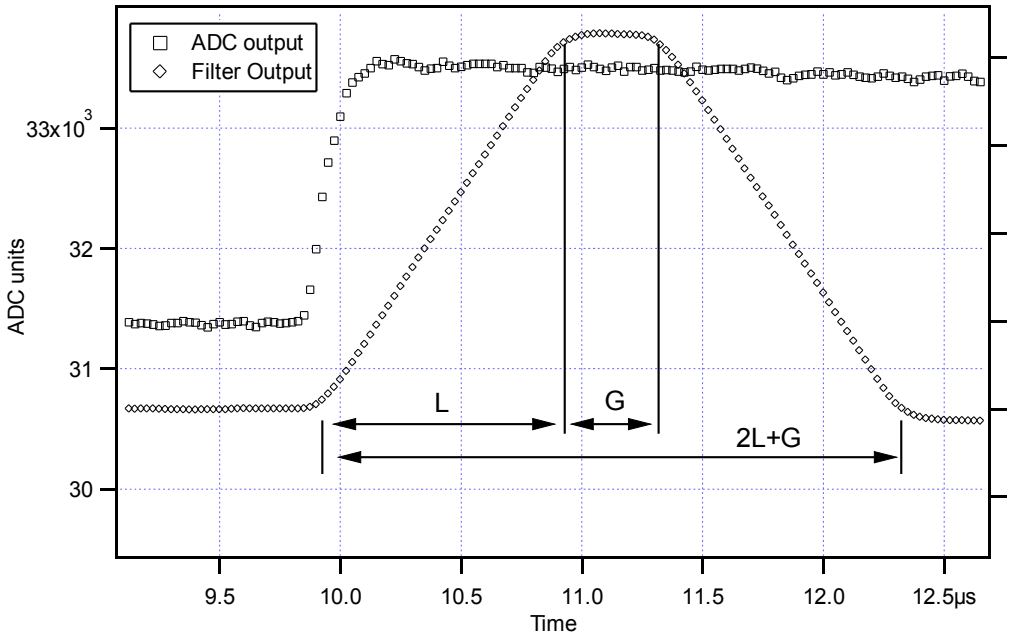


Figure 6.3: Trapezoidal filtering of a preamplifier step with L=1µs and G=0.4µs.

6.3 Baselines and Preamplifier Decay Times

Figure 6.4 shows an event over a longer time interval and how the filter treats the preamplifier noise in regions when no γ-ray pulses are present. As may be seen the effect of the filter is both to reduce the amplitude of the fluctuations and reduce their high frequency content. This region is called the *baseline* because it establishes the reference level from which the γ-ray peak amplitude V_x is to be measured. The fluctuations in the baseline have a standard deviation σ_e which is referred to as the *electronic noise* of the system, a number which depends on the rise time of the filter used. Riding on top of this noise, the γ-ray peaks contribute an additional noise term, the *Fano noise*, which arises from statistical fluctuations in the amount of charge Q_x produced when the γ-ray is absorbed in the detector. This Fano noise σ_f adds in quadrature with the electronic noise, so that the total noise σ_t in measuring V_x is found from

$$\sigma_t = \text{sqrt}(\sigma_f^2 + \sigma_e^2) \quad (3)$$

© XIA 2012. All rights reserved.

The Fano noise is only a property of the detector material. The electronic noise, on the other hand, may have contributions from both the preamplifier and the amplifier. When the preamplifier and amplifier are both well designed and well matched, however, the amplifier's noise contribution should be essentially negligible. Achieving this in the mixed analog-digital environment of a digital pulse processor is a non-trivial task, however.

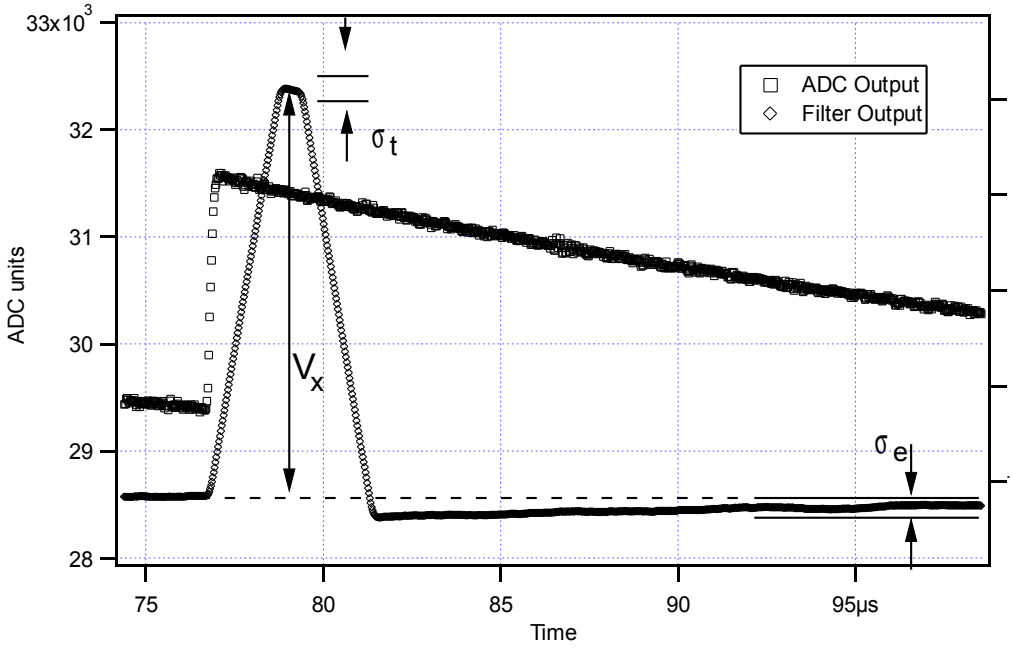


Figure 6.4: A γ-ray event displayed over a longer time period to show baseline noise and the effect of preamplifier decay time.

With a RC-type preamplifier, the slope of the preamplifier is rarely zero. Every step decays exponentially back to the DC level of the preamplifier. During such a decay, the baselines are obviously not zero. This can be seen in Figure 6.4, where the filter output during the exponential decay after the pulse is below the initial level. Note also that the flat top region is sloped downwards.

Using the decay constant τ, the baselines can be mapped back to the DC level. This allows precise determination of γ-ray energies, even if the pulse sits on the falling slope of a previous pulse. The value of τ, being a characteristic of the preamplifier, has to be determined by the user and host software and downloaded to the module.

6.4 Thresholds and Pile-up Inspection

As noted above, we wish to capture a value of V_x for each γ-ray detected and use these values to construct a spectrum. This process is also significantly different between digital and analog systems. In the analog system the peak value must be "captured" into an analog storage device, usually a capacitor, and "held" until it is digitized. Then the digital value is used to update a memory location to build the desired spectrum. During this analog to digital conversion process the system is dead to other events, which can severely reduce system throughput. Even single channel analyzer systems introduce significant deadtime at this stage since they must wait some

© XIA 2012. All rights reserved.

period (typically a few microseconds) to determine whether or not the window condition is satisfied.

Digital systems are much more efficient in this regard, since the values output by the filter are already digital values. All that is required is to take the filter sums, reconstruct the energy V_x, and add it to the spectrum. In the Pixie-4, the filter sums are continuously updated by the RTPU (see section 5.2), and only have to be read out by the DSP when an event occurs. Reconstructing the energy and incrementing the spectrum is done by the DSP, so that the RTPU is ready to take new data immediately after the readout. This usually takes much less than one filter rise time, so that no system deadtime is produced by a "capture and store" operation. This is a significant source of the enhanced throughput found in digital systems.

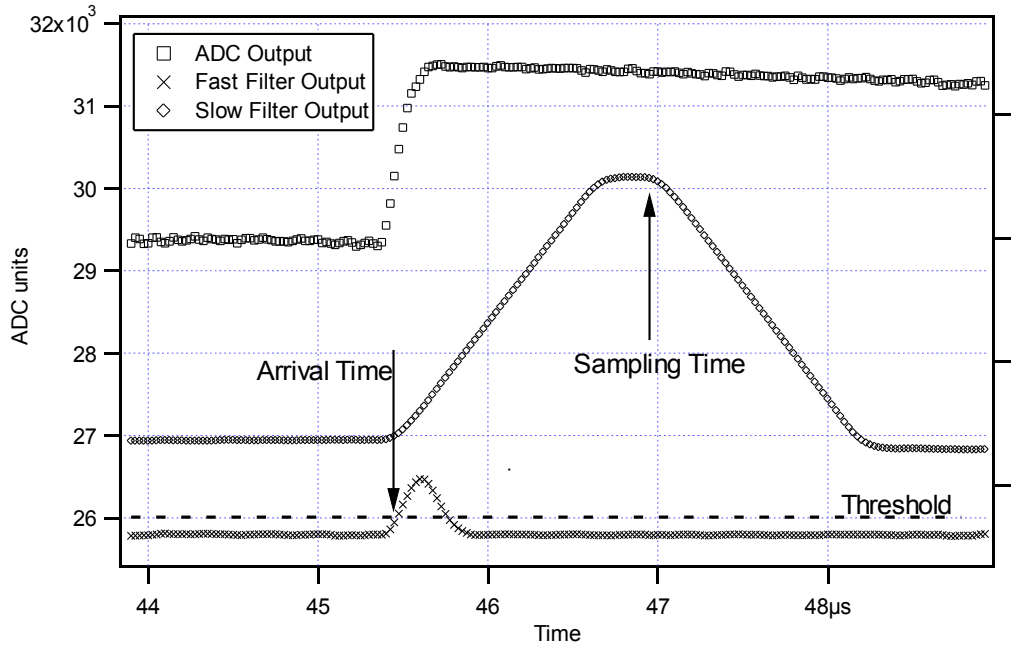


Figure 6.5: Peak detection and sampling in the Pixie-4.

The peak detection and sampling in the Pixie-4 is handled as indicated in Figure 6.5. Two trapezoidal filters are implemented, a *fast filter* and a *slow filter*. The fast filter is used to detect the arrival of γ-rays, the slow filter is used for the measurement of V_x, with reduced noise at longer filter rise times. The fast filter has a filter length $L_f = 0.1\mu s$ and a gap $G_f = 0.1\mu s$. The slow filter has $L_s = 1.2\mu s$ and $G_s = 0.35\mu s$.

The arrival of the γ-ray step (in the preamplifier output) is detected by digitally comparing the fast filter output to THRESHOLD, a digital constant set by the user. Crossing the threshold starts a counter to count PEAKSAMP clock cycles to arrive at the appropriate time to sample the value of the slow filter. Because the digital filtering processes are deterministic, PEAKSAMP depends only on the values of the fast and slow filter constants and the rise time of the preamplifier pulses. The slow filter value captured following PEAKSAMP is then the slow digital filter's estimate of V_x.

© XIA 2012. All rights reserved.

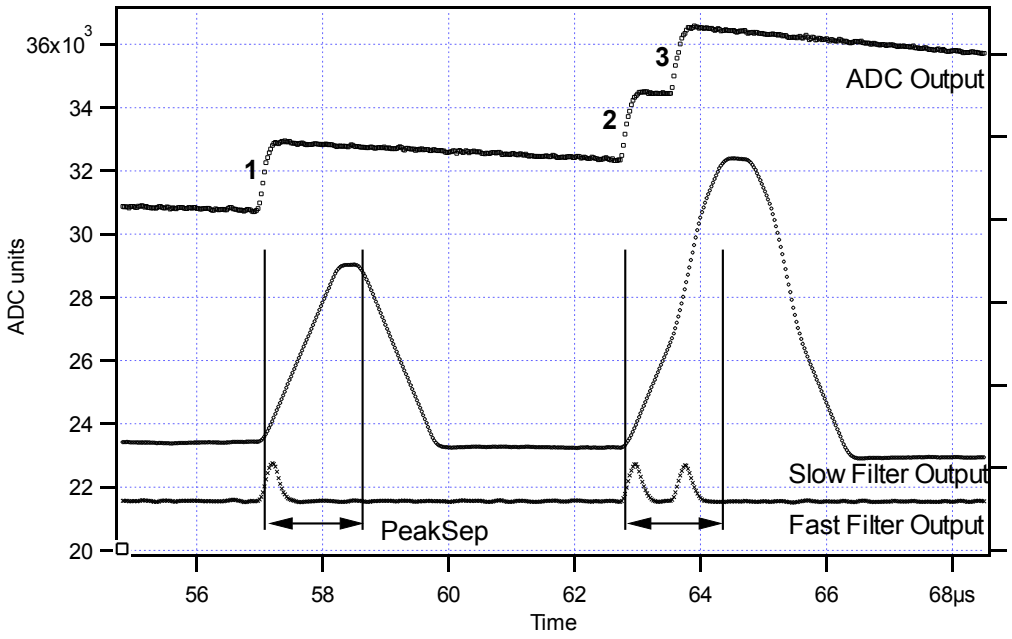


Figure 6.6: A sequence of 3 γ-ray pulses separated by various intervals to show the origin of pileup and demonstrate how it is detected by the Pixie-4.

The value V_x captured will only be a valid measure of the associated γ-ray's energy provided that the filtered pulse is sufficiently well separated in time from its preceding and succeeding neighbor pulses so that their peak amplitudes are not distorted by the action of the trapezoidal filter. That is, if the pulse is not *piled up*. The relevant issues may be understood by reference to Figure 6.6, which shows 3 γ-rays arriving separated by various intervals. The fast filter has a filter length $L_f = 0.1\mu s$ and a gap $G_f = 0.1\mu s$. The slow filter has $L_s = 1.2\mu s$ and $G_s = 0.35\mu s$.

Because the trapezoidal filter is a linear filter, its output for a series of pulses is the linear sum of its outputs for the individual members in the series. Pileup occurs when the rising edge of one pulse lies under the peak (specifically the sampling point) of its neighbor. Thus, in Figure 6.6, peaks 1 and 2 are sufficiently well separated so that the leading edge of peak 2 falls after the peak of pulse 1. Because the trapezoidal filter function is symmetrical, this also means that pulse 1's trailing edge also does not fall under the peak of pulse 2. For this to be true, the two pulses must be separated by at least an interval of L + G. Peaks 2 and 3, which are separated by less than 1.0 µs, are thus seen to pileup in the present example with a 1.2 µs rise time.

This leads to an important point: whether pulses suffer slow pileup depends critically on the rise time of the filter being used. The amount of pileup which occurs at a given average signal rate will increase with longer rise times.

Because the fast filter rise time is only 0.1 µs, these γ-ray pulses do not pileup in the fast filter channel. The Pixie-4 can therefore test for slow channel pileup by measuring the fast filter for the interval PEAKSEP after a pulse arrival time. If no second pulse occurs in this interval, then there is no trailing edge pileup and the pulse is validated for acquisition. PEAKSEP is usually set to a value close to L + G + 1. Pulse 1 passes this test, as shown in Figure 6.6. Pulse 2, however, fails the PEAKSEP test because pulse 3 follows less than 1.0 µs. Notice, by the symmetry of the

© XIA 2012. All rights reserved.

trapezoidal filter, if pulse 2 is rejected because of pulse 3, then pulse 3 is similarly rejected because of pulse 2.

6.5 Filter Range

To accommodate the wide range of filter rise times from 0.053 μs to 106 μs, the filters are implemented in the RTPUs using FPGA configurations with different clock decimations (filter ranges). The ADC sampling rate is always 13.3ns, but in higher clock decimations, several ADC samples are averaged before entering the filtering logic. In filter range 1, 2^1 samples are averaged, 2^2 samples in filter range 2, and so on. Since the sum of rise time and flat top is limited to 127 decimated clock cycles, filter time granularity and filter time are limited to the values listed in Table 6.1.

Table 6.1: RTPU clock decimations and filter time granularity

Filter range	Filter granularity	max. $T_{rise}+T_{flat}$	min. T_{rise}	min. T_{flat}
1	0.027μs	3.387μs	0.053μs	0.08μs
2	0.053μs	6.773μs	0.107μs	0.16μs
3	0.107μs	13.547μs	0.213μs	0.32μs
4	0.213μs	27.093μs	0.427μs	0.64μs
5	0.427μs	54.187μs	0.853μs	1.28μs
6	0.853μs	108.373μs	1.707μs	2.56μs

All filter ranges are implemented in the same FPGA configuration. Only the "FILTERRANGE" parameter of the DSP has to be set to select a particular filter range.

6.6 Dead Time and Run Statistics

6.6.1 Definition of dead times

Dead time in the Pixie-4 data acquisition can occur at several processing stages. For the purpose of this document, we distinguish three types of dead time, each with a number of contributions from different processes.

6.6.1.1 Dead time associated with each pulse

1. Filter dead time

At the most fundamental level, the energy filter implemented in the FPGA requires a certain amount of pulse waveform (the "filter time") to measure the energy. Once a rising edge of a pulse is detected at time T0, the FPGA computes three filter sums using the waveform data from T- (a energy filter rise time before T0) to T1 (a flat top time plus filter rise time after T0), see section 6.4 and figure 6.7. If a second pulse occurs during this time, the energy measurement will be incorrect. Therefore, processing in the FPGA includes pileup rejection which enforces a minimum distance between pulses and validates a pulse for recording only if no more than one pulse occurred from T0 to T1 (in the previous firmware, T- to T1). Consequently, each pulse creates a dead time Td = (T1 – T0) equal to the filter time. This dead time, simply given by the

© XIA 2012. All rights reserved.

time to measure the pulse height, is unavoidable unless pulse height measurements are allowed to overlap (which would produce false results).

Assuming randomly occurring pulses, the effect of dead time on the output count rate is governed by Poisson statistics for paralyzable systems with pileup rejection[7]. This means the output count rate OCR (valid pulses) is a function of filter dead time Td and input count rate ICR given by

$$OCR = ICR * \exp(-ICR* 2 * Td), \quad (4)$$

which reaches a maximum $OCR_{max} = ICR_{max}/e$ at $ICR_{max} = 1/(2*Td)$. Simply speaking, the factor 2 for Td comes from the fact that not only is an event E2 invalid when it falls into the dead time of a previous event E1, but E1 is rejected as piled up as well.

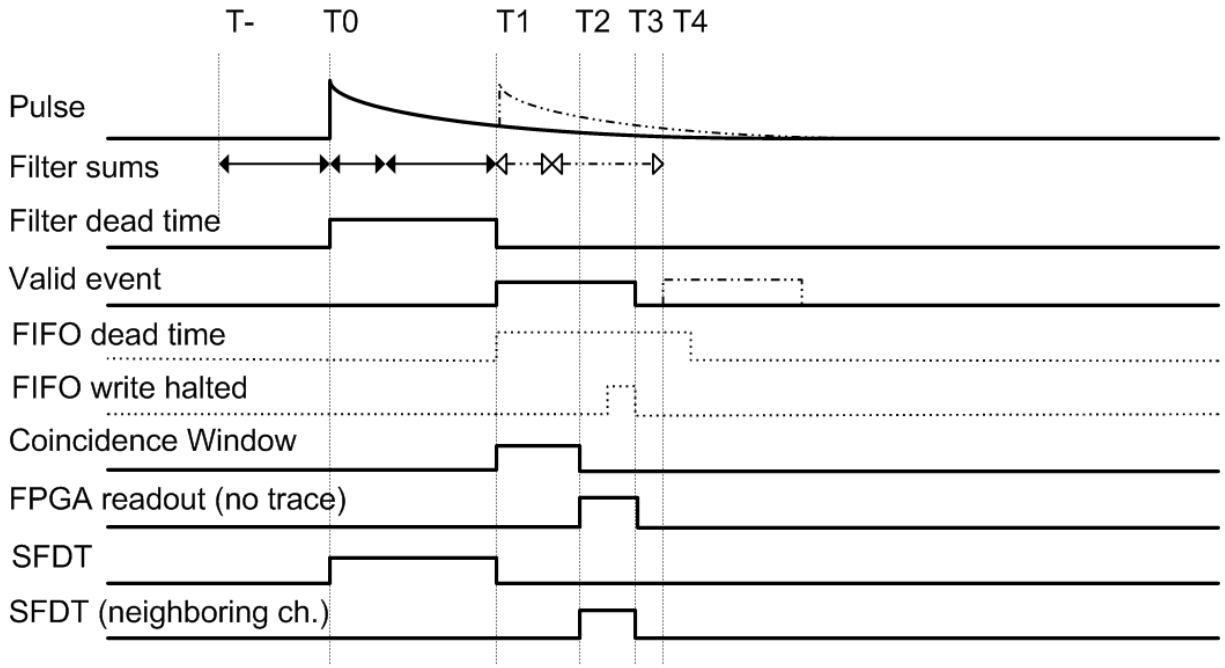


Fig. 6.7. Pulse dead time without trace capture. A second pulse arriving at T1+x will pass pileup inspection at time T4+x. No dead time is incurred for the FPGA readout if it is completed at a time T3 before T4 (T4-T1 = T1-T0, the filter dead time). The FIFO dead time is ignored.

As a consequence of the pileup inspection, there is a delay of one filter time between the rising edge of the pulse and the decision to record the data, i.e. the validation as not piled up. This delay can be used to pipeline the subsequent processing steps: the coincidence window and FPGA readout of a first pulse can overlap with the pileup inspection of a second pulse, as described below.

2. FIFO dead time

The second unavoidable dead time comes from the waveform capture FIFO. When the user requires waveform data from before time T0, the FIFO must contain this pre-trigger data at time T0, else the event data is incomplete and the event is rejected. This means that whenever the

[7] G. Knoll, Radiation and Measurement, J Wiley & Sons, Inc, 2000, chapters 4 and 17.

© XIA 2012. All rights reserved.

FIFO write process was stopped, it takes a pre-trigger time to refresh the pre-trigger data and be ready for new events. To minimize this effect, the FIFO write process is stopped only in two cases, when it is necessary to avoid overwriting potentially valid data: a) When the event validation takes longer than the time available in the FIFO (13.6 μs minus the pre-trigger time) and b) when the readout of a valid event by the DSP is not completed before the FIFO is filled. In the current firmware, the impact of case b) is practically eliminated by restarting the FIFO write process before the DSP readout is finished, overwriting data already read with fresh pre-trigger data. (3 writes can be performed for one read, and the writing is resumed after ¾ of the waveform is read)

In the current firmware implementation, the FIFO logic does not allow the post-trigger data of a second pulse to be stored while a first pulse is waiting for readout. This means if waveforms are requested, overlap of a second pulse with the FPGA readout is not possible because the waveform data of the second pulse can not be recorded. Essentially the FIFO is dead for new events until the FPGA readout is complete. This effect is usually more significant than the cases a) and b) above, at least for systems like HPGe detectors and scintillators where the filter time (typically 2-6 μs) is well below the FIFO limit of 13.6 μs minus the pre-trigger time.

In any case, the FIFO dead time from a) and b) and the FIFO's effect to prevent overlap of a second pulse with the FPGA readout is only in effect when waveforms are required by the user, i.e only in list mode runs with non-zero tracelengths. It is therefore important to set the tracelength to zero in compressed list mode runs (0x101-103) if no pulse shape analysis is performed.

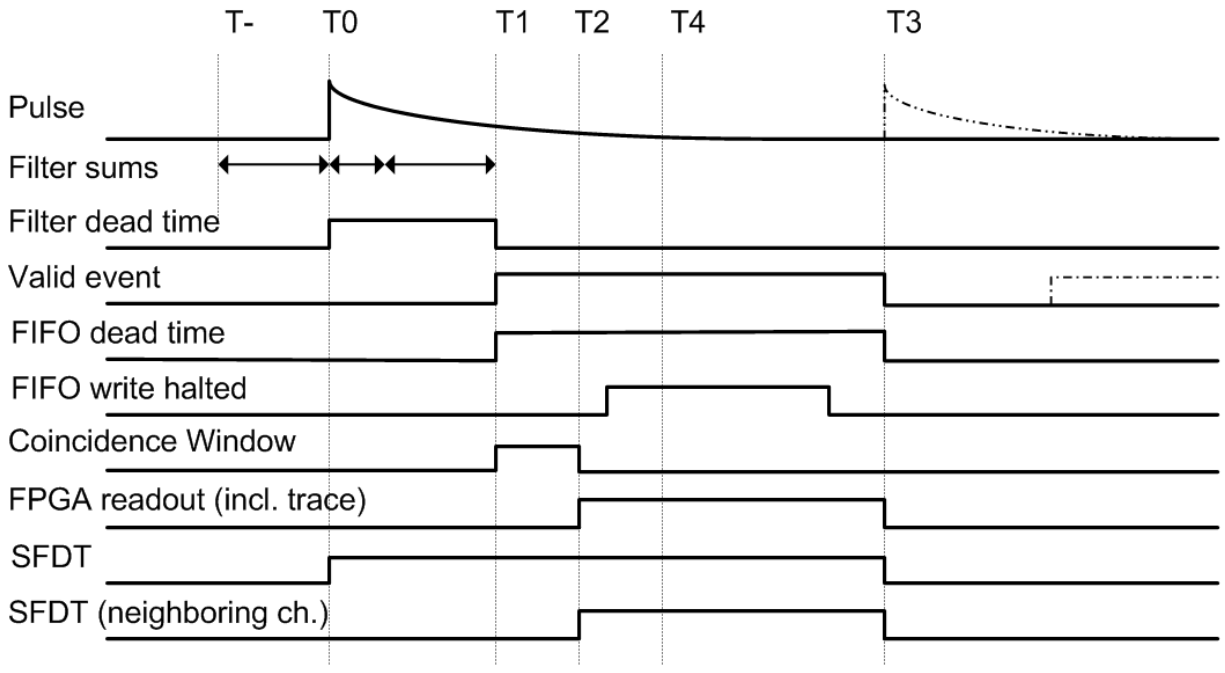


Fig. 6.8. Pulse dead time with trace capture. The FIFO blocks new events as long there is a valid event in the FPGA. Therefore the soonest arrival time for a second pulse is at T3 after the FPGA readout is complete (now later than T4). The FIFO write resumes before T3, overwriting data already read out, and thus contains pre-trigger data for the second pulse at T3.

© XIA 2012. All rights reserved.

3. Coincidence window

Each valid event is further subject to a coincidence test, i.e. checking the hit pattern against the user defined pattern of acceptable events. The test itself requires about 5 clock cycles overhead plus a minimum window of 1 clock cycle and thus adds about 80 ns of dead time. Any time added by the user to the coincidence window increases the dead time accordingly. However, the full dead time is only incurred by the first channel that is validated, since the coincidence window is counted from the first event validation. For example, if a second channel sees no pulse during the coincidence window, it will be active during the full coincidence window. If a third channel sees a pulse and is validated by passing its local pileup inspection 500ns after the first, it was active for 500 ns longer than the first channel and incurred dead time of (coincidence window – 500ns). Similar to the FPGA readout described below, the coincidence window may overlap with the pileup inspection of a subsequent pulse, so it only adds to dead time if FPGA readout and coincidence window combined are longer that the filter time.

4. FPGA readout dead time

The current firmware only allows valid data of one event at a time in the FPGA. The DSP reads out the data after receiving an event interrupt if the coincidence test was passed. Thus once an event is validated, the channel may incur dead time until the DSP finishes the data readout of hit pattern, energy filter sums, and possibly waveforms. However, the pileup inspection of a subsequent pulse can already begin while the first event is read out, as long as it finishes after the DSP has completed the readout. This means the DSP has a filter time (minus coincidence window) available for readout. The readout creates additional dead time only if it exceeds this time. For typical HPGe filter settings with no traces and minimum coincidence window, the FPGA readout time usually fulfills this requirement and thus introduces no additional dead time.

However, when traces are required, the FIFO logic does not allow the post-trigger data of a second pulse to be stored until the readout is completed, so in this case the readout always creates dead time equal to the full readout time. Details are as follows:

a) Hit pattern readout

At the beginning of the readout, the DSP reads the hit pattern and determines which channel to read. This, together with some general overhead, takes about 850 ns.

b) filter sum readout

Readout of filter sums takes about 600 ns per channel marked in the hit pattern.

c) waveform readout

Waveform readout occurs in the same processing step as the filter sum readout and increases the DSP readout time by 3x the recorded waveform length. It can be avoided by setting the requested waveform length to zero. Note that if the waveform length is nonzero, waveforms are still read in the "compressed" run types 0x101-103 for possible pulse shape analysis, only the storing of waveforms in the output data stream is disabled. In MCA runs, waveforms are never read, so the waveform length is irrelevant.

In the current firmware, the FPGAs are released to resume data acquisition only after all channels marked in the hit pattern have been read out. At this time, all channels are cleared after

© XIA 2012. All rights reserved.

readout, which means that pulses validated after the end of the coincidence window (and thus not contributing to the hit pattern) are lost. The readout therefore causes dead time in all other channels in the module as well.

5. Fast trigger dead time (FTDT)

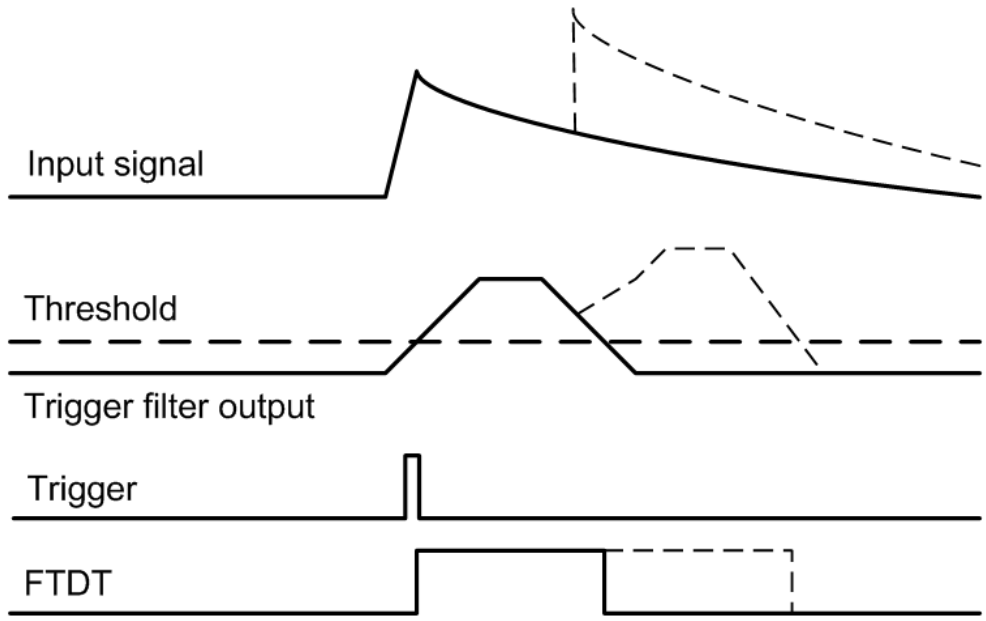


Fig. 6.9. Fast Trigger Dead Time (FTDT). A second pulse is not detected if the trigger filter output is still above threshold.

An additional type of dead time only affects the trigger filter. Triggers are issued when the trigger filter output goes above the trigger threshold set by the user. However, depending on the length of the trigger filter and the rise time of the input signal, the trigger filter output will remain above threshold for a finite amount of time. During this time, no second trigger can be issued[8]. Therefore triggers are not counted during this time, and when computing the input count rate, the time lost has to be taken into account. FTDT is thus purely a correction for the computation of the input count rate.

6. Summary

As listed in Table 6.2, the dead time associated with each pulse is essentially in the order of the filter dead time, unless waveforms are requested by the user. Even with waveforms, few counts are lost at low count rates for short waveforms. Losses are significant with longer waveforms or at higher count rates. To some extend, this is due to the general purpose FIFO logic which allows waveforms up to the maximum available memory in the FPGA. For application where only short traces are required, lower dead times are possibly by reorganizing memory to buffer multiple events in the FPGA. Please contact XIA for details.

[8] The MAXWIDTH parameter can be used to define a maximum acceptable time over threshold and thus to reject events piled up "on the rising edge".

© XIA 2012. All rights reserved.

Table 6.2 Examples of dead times in typical conditions. Assumes events with only one active channel.

	MCA run, 2µs filter, any ICR	**MCA run, 6µs filter, any ICR**	**LM run, 6µs filter, no trace, ICR = 1k/s**	**LM run, 6µs filter, 2µs trace, ICR = 1k/s**	**LM run, 6µs filter, 2µs trace, ICR = 10k/s**
A: Filter dead time	1 µs	6 µs	6 µs	6 µs	6 µs
B: FIFO dead time[1]	--	--	--	C + D + 0us (write not stopped)	C + D + 0us (write not stopped)
C: Coinc. Window	0.133 µs	0.133 µs	0.133 µs	0.133 µs	0.133 µs
D: FPGA readout	1.5 µs	1.5 µs	1.5 µs	1.5 µs +3 x 2 µs = 7.5 µs	1.5 µs +3 x 2 µs = 7.5 µs
SFDT per event (= Td)	max(A,B+C) = 1.633 µs	max(A,B+C) = 6 µs	max(A,B+C) = 6 µs	A+B (B includes C,D) = 13.633 µs	A+B (B includes C,D) = 13.633 µs
OCR	ICR * $e^{(-ICR* 2 * Td)}$		988/s	973/s	7514/s
Fraction events lost	$1-e^{(-ICR* 2 * Td)}$		1.2%	2.7%	24%

Note: 1. FIFO logic prevents overlap of second pulse with coincidence window and FPGA readout, but only when waveforms are requested. Additional dead time due to stopped FIFO write process occurs only when filter dead time > 13.6 µs – pre-trigger time

6.6.1.2 Dead time associated with external conditions

There are three dead time effects that originate from outside the trigger/filter FPGA. The first two have the effect of stopping the Pixie-4 live time counter, the last is counted separately.

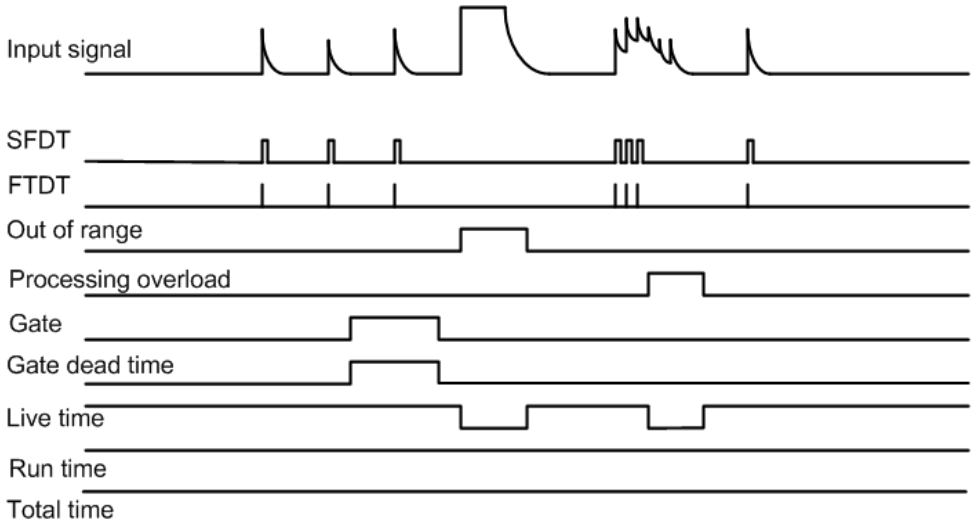


Fig. 6.10. The live time counter is stopped when the signal is out of range and when events are rejected because of a processing backlog in the DSP or spectrum memory increment process in the FPGA. SFDT and FTDT are only counted when the live time is on. The gate dead time is counted in

© XIA 2012. All rights reserved.

a separate counter, but also only when the live time is on. Run time and total time are always on unless the run is stopped (see below).

1. Signal out of range

When detector gains or offsets drift, or an unusual large pulse is generated in the detector, the analog input of the ADC may go out of range. In this condition, the FPGA can not accumulate meaningful filter sums and thus is considered dead. This dead time is enforced during the actual out-of-range condition and several filter times afterwards until the bad ADC samples are purged from filter memory. The live time counter is stopped during the out-of-range condition because no triggers can be issued and no pulses are counted.

2. On-board pulse processing limit

Processing dead time includes all time lost in the acquisition due to the conversion of the raw energy filter sums into the pulse height which is stored in list mode memory and/or binned into spectrum memory. This conversion process begins after the raw filter sums have been read out from the FPGA. The FPGA resumes its acquisition after the readout and raw filter sums for several hundred events can be buffered in DSP memory. Therefore the computation of the pulse height in the DSP incurs no dead time as long as a) average count rates are below the maximum DSP processing rate and b) bursts of pulses do not fill the raw data buffer faster than the DSP can process it.

a) The maximum processing rate depends on the multiplicity of the event and whether pulse shape analysis is performed on the waveforms. For standard processing as in MCA runs, it is about 240,000-480,000 pulses/s, i.e. the DSP spends roughly 2-4 μs per pulse to compute the energy. (Processing events containing pulses from several channels has less overhead and is thus faster per pulse.) This rate is much higher than the maximum throughput set by Poisson statistics for most typical filter times.

b) The size of the raw data buffer is 8K words in mode 0x100, else about 4K words. An event without waveforms, consisting of 1-4 pulses from the 4 channels of a module, contains 9-12 raw data words per pulse (depending on the number of channels contributing). This means up to ~400 events can be stored in the raw buffer, and it does not matter how closely they follow each other as long as on average, the DSP can keep up with the processing. For the standard processing in MCA runs as above, the time to process a pulses is about 2-4 μs. Therefore, no events are lost if for example the 400 pulses occur in 400 μs followed by a pause of 400-1200 μs (i.e. 1M/s bursts with a duty cycle of 50-25%) or in 200 μs followed by a pause of 600-1400 μs (2M/s bursts with a duty cycle of 25-12.5%), and so on.

   However, if waveforms are read and pulses contain more raw data words, the raw buffer can contain correspondingly fewer events. For example, when collecting 2 μs waveforms, a pulse consists of ~160 words and thus there is only room for ~25 pulses. Assuming no pulse shape analysis, the time to process a pulse is again roughly 2-4 μs. Thus the buffering limit changes from less than 400 pulses in ~800-1600 μs to less than 25 pulses in 50-200 μs. Obviously, the average rate stays the same, but the length of bursts is reduced.

   In any case, burst rates are still limited by Poisson statistics (filter dead time) and the FPGA readout time, if it exceeds the filter time or if waveforms are read.

© XIA 2012. All rights reserved.

3. Gating or Veto (GFLT)

If an external signal prohibits acquisition using the GATE or VETO signals, the channel is also dead (though on purpose). As further described in section 7.4, the use of these signals may depend on the application:

a) On one hand they may be used to reject an individual pulse (e.g. externally summing multiplicities from several channels and issuing a short validation pulse at the right time in the validation process). In this case the actual length of the pulse does not correspond to a dead time. The VETO input is set up for this purpose and we call this mode of operation GFLT (global first level trigger for validation).

b) On the other hand GATE or VETO may block validation of events for certain amounts of time (e.g. changing sources or activating beams). In this case they should be counted clock cycle by clock cycle as dead time. Both the VETO and the GATE inputs are available for this purpose, VETO as a global signal for the whole system, GATE as a dedicated signal for each channel. VETO acts at the time of pulse validation, GATE acts at the time of the rising edge of the pulse. However, the VETO input can be routed to replace the GATE input with a user option.

c) In a third class of application, the acquisition may only be of interest when GATE or VETO are off. The pulse rejection logic would be similar to b), but livetimes and count rates should only be counted when GATE/Veto are off as count rates would be different in on and off times. (In b) one would be more interested in an overall livetime and average count rate and additionally the time inhibited by GATE or VETO to make corrections.)

The appropriate way to count GATE or VETO dead time may thus depend on the experiment. See below (GDT) for the current methods implemented in the firmware.

6.6.1.3 Dead time associated with host readout

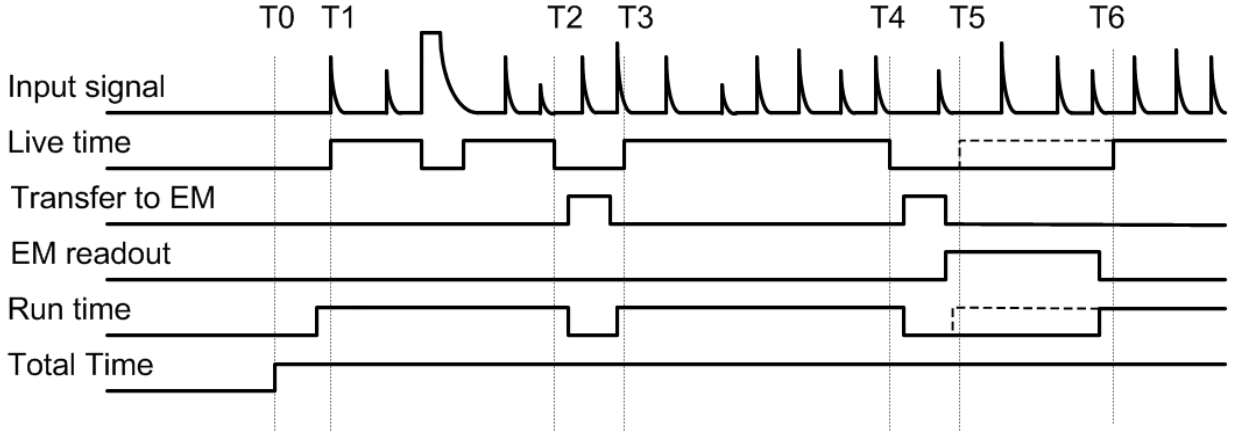


Fig. 6.11 After a run start command arrives from the host at T0, the Pixie-4 module acquires several buffers of data (T1-T2, T3-T4) until the external memory (EM) is full and is read out by the host at T5. (For single buffer mode not using external memory, the transfer is replaced by a host readout, which however takes much longer.)

© XIA 2012. All rights reserved.

The final type of dead time comes from the readout of data from Pixie-4 memory to the host PC. In MCA mode, this is limited to the access arbitration for the spectrum memory. The memory has only a single port for both the increments according to the pulse height computed by the DSP and for readout to the host PC, arbitrated by an FPGA. While the host is reading the memory, spectrum increments are queued in a buffer (2K long). At maximum count rate, it will take the DSP at least (2K * processing time) to fill the buffer and correspondingly longer at lower count rates, while the host readout typically takes ~30 ms. Thus host readout dead time is usually not an issue in MCA runs unless spectra are read very frequently.

In list mode runs, the Pixie-4 memory fills up after a certain number of events are acquired. Acquisition is stopped until the memory is read out. Depending on the "buffer per spill" setting, this is organized in one of three ways:

a) In single buffer mode, acquisition stops after the DSP's 8K buffer is filled and data is read out in a slow transfer mode.

b) In 32 buffer mode, the 8K buffer is automatically transferred to external memory. The transfer incurs about 300µs dead time (this may be improved in the future). After 32 transfers, the external memory is full and acquisition is stopped. The external memory is read out in a fast block transfer that takes about 30 ms

c) In 16/16 buffer mode (or double buffer mode), the 8K buffer is also automatically transferred to external memory, but only 16 times. Then a flag is raised so that the host can read out the memory in a fast block transfer while acquisition continues and new data is stored in the second half of the memory. However, since again the memory has only a single port, host readout can not happen at the same time as the DSP transfer. Normally, the readout will be finished before the DSP 8K buffer is filled, but at high count rate or when recording waveforms, the transfer may have to wait until the host finished reading the external memory, which means the acquisition is stopped and there will be dead time in addition to the ~300 us transfer time.

Case b) is pictured in Fig. 6.11; the difference for case c) is shown with the dashed lines. The Pixie-4 acquires several buffers of data (T1-T2, T3-T4) until the external memory is full and read out by the host at T5. After readout, a second spill resumes at T6. The live time counter is active while events can be acquired, which excludes memory transfer and readout. The run time starts a bit earlier and lasts a bit longer than the live time for each buffer, and does not switch off for out-of range or other conditions. The total time counts all time from T0 to the end of the last spill. The Pixie 4 can not count the time between the user clicking the run start button and T0. In case c), the acquisition resumes already at T5 while the memory is read out independently. In a) and b), there is thus dead time in the Pixie-4 while it is waiting for the host to read out the data. In b) and c) there is dead time while the data is transferred to external memory. In all cases, the run is considered stopped in the Pixie-4, as opposed to the dead times described in section 6.6.1.2. The readout dead times are omitted from the Pixie-4 live time and run time counters (counters are stopped), but included in the "total time" counter.

Examples of host readout dead times are shown in Table 6.3. Rows A and B are copied from Table 6.2 for comparison. When no traces are recorded, the fraction of time lost to readout is less than 1% even up to moderately high count rates; and well below the fraction of events lost due to Td (dominated by the filter time). When traces are recorded, the fraction of time lost increases, but is still below the fraction of events lost due to Td.

© XIA 2012. All rights reserved.

Table 6.3. Dead times from host readout in 32 buffer/spill mode. (One channel active per event)

	LM run 0x103, 6µs filter, no trace, ICR = 1k/s	LM run 0x103, 6µs filter, no trace, ICR = 10k/s	LM run 0x100, 6µs filter, no trace, ICR = 1k/s	LM run 0x100, 6µs filter, 2µs trace, ICR = 1k/s	LM run 0x100, 6µs filter, 2µs trace, ICR = 10k/s
A: OCR	988/s	8869/s	988/s	973/s	7614/s
B: Fraction events lost to Td (from table 6.2)	1.2%	11.3%	1.2%	2.7%	24%
C: Events in buffer (8K / Nwords)	1637	1637	682	50	50
D: Buffer fill time = C / A	1.657 s	0.185 s	0.690 s	51.4 ms	6.5 ms
E: EM fill time = 32 x (D + 0.30ms)	53.034 s	5.930 s	22.090 s	1.654 s	0.218 s
F: Total readout time = (32x 0.55 +30)ms ~ TT-LT per spill	39.6ms	39.6ms	39.6ms	39.6ms	39.6ms
Fraction time lost to readout = F/(E+30ms)	<0.1%	0.7%	0.2%	2.3%	16.0%

6.6.2 Live and dead time counters

The Pixie-4 firmware has been optimized to reduce the dead time as much as possible, and a number of counters measure the remaining dead times as well as the number of counts to provide information for dead time correction. The result of these counters is stored in the following DSP output variables:

TOTAL TIME

The TOTAL TIME is an attempt to measure the real laboratory time during which the Pixie-4 module was requested to take data. It essentially counts the time from the most recent command to start a new run, i.e. in list mode runs the start of the first spill. The TOTAL TIME includes the time spent for run start initialization and host readout. Thus it can be used together with the LIVE TIME to determine the fraction of the real time the module was actively taking data. However, since it is based on the Pixie-4's internal oscillator (a part with 50 ppm accuracy from module to module), it may not be as precise as a "laboratory wall clock" over long time spans (e.g. the host PC's internal clock). Also, it does not take into account the time required to send commands from the PC to the module. For the "DAQ Fraction" displayed in the Pixie Viewer, the PC's time is used.

RUN TIME

The RUN TIME variable gives the time during which the DSP on the Pixie-4 module was "switched on" for data acquisition. The usefulness of this variable is very limited. It is less than the real time passed in the laboratory during acquisition (e.g. the TOTAL TIME or the time measured by the host PC) because it omits the time for a) start run signals from the PC to reach

© XIA 2012. All rights reserved.

the DSP, b) communication between modules to synchronize the run start, c) clearing of memory and output variables before starting a run, and d) time spent waiting for readout. It is larger than the time the FPGA is ready to take data because it does not account for the dead time from filter, FIFO, or DSP readout. Thus for most purposes, the LIVE TIME, TOTAL TIME, or the host PC's real time (not the DSP variable named REAL TIME) are more appropriate. This variable is provided mainly to provide backwards compatibility to previous software revisions.

LIVE TIME

The LIVE TIME is counted in the FPGA independently for each channel and measures the time the channel is ready for acquisition. The LIVE TIME counter starts when the DSP finished all setup routines at the beginning of a run, omits the times the ADC signal is out of range or the DSP can not keep up with processing (section 6.6.1.2, 1. and 2.), and ends when the DSP encounters an end run condition (e.g. memory full or host stop). Subsequent spills add in the same way. It is thus the time during which triggers are counted and can cause recording (or pile up) of data, the best available measurement of the time the channel was active.

FTDT (fast trigger dead time)

The fast trigger dead time counts the time the trigger filter is unable to issue triggers because the trigger filter output is already above threshold (and can not recognize a second pulse). It does not include the time triggers have been "paused" for a short time after a first trigger (an advanced user option to suppress double triggering), because the concept is that all triggers occurring during the pause are counted as only one trigger. When computing the input count rate, one should divide the number of triggers counted (FASTPEAKS) by the difference (LIVE TIME – FTDT) since triggers are not counted during FTDT.

SFDT (slow filter dead time)

The slow filter dead time counts the time new triggers will not lead to the recording of new data. This includes effects 1-4 listed in section 6.6.1.1 as dead time associated with a pulse. In detail,

- it includes the time the pileup inspection is taking place and the summation of energy filter sums is in progress,
- it includes the time the FIFO does not contain fresh pre-trigger data or is unable to accept a new event
- it includes the time from event validation until one filter time before the end of the DSP readout. This includes DSP readout of an event in a different channel after which data in all channels are discarded. (the end of dead time counting before the end of the DSP readout is implemented as a delayed start of the counting)

GDT (GATE dead time)

The dead time from GATE or VETO/GFLT is counted separately from SFDT for each channel. As mentioned above and further described in section 7.4, the use of these signals may depend on the application.

In the current firmware, a rising (or optionally falling) edge at the Gate input creates a GATE PULSE with length GATE WINDOW (see section 7.4). Both this GATE PULSE and the (optionally inverted) signal on the VETO input are combined into GDT_ON. While GDT_ON is high, GDT is incremented every clock cycle and thus counts the time pulses can be rejected from

© XIA 2012. All rights reserved.

acquisition. (The rejection has to be enabled independently). It is possible to instead count the time during which pulses are allowed by inverting the combined signal. In the "standard statistics" mode, the GDT counter is additionally subject to the channel being live, i.e. GDT is only counted if a run is in progress, signal in range, etc.

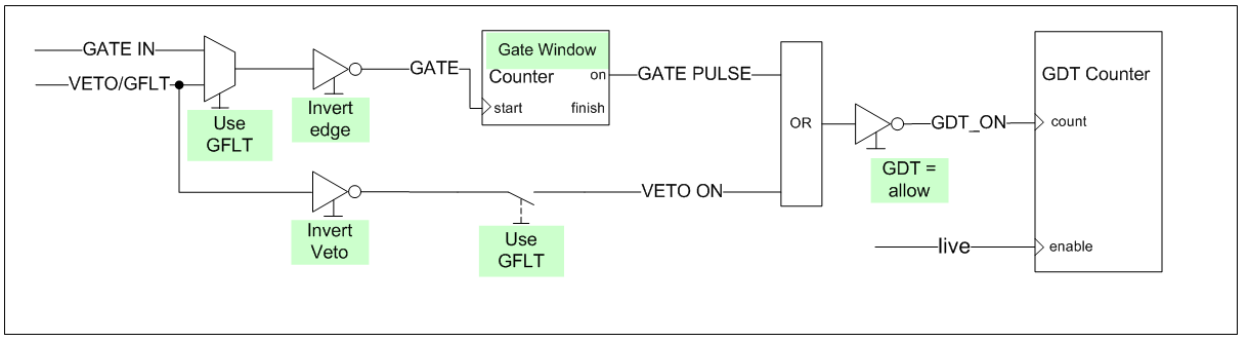


Fig. 6.12 GDT counting logic shown for standard statistics mode. In Gate statistics mode, the channel LIVE signal is ignored for counting GDT and in turn LIVE is subject to GDT_ON.

To support the third case of gating mentioned above, where the acquisition is only of interest when GATE or VETO are off, there is an alternate "GATE statistics" mode to count GDT and livetimes. In Gate statistics mode, all time and rate counters except RUN TIME and TOTAL TIME are only active if GDT_ON is high. This means GATE or VETO must be present for the channel to be live. In turn, GDT is counted independently from whether the module is live or not.

If the VETO input is unused but defaults to high so that GDT is equal to the live time, invert the VETO polarity in the CHANNEL REGISTER Panel to only count the GATE PULSE time. The VETO input can be used instead of the GATE input for the gating circuitry, and if this option is used only the GATE PULSE derived from the VETO input is counted, not the original VETO.

For the case that the Veto input is used for a GFLT-type validation pulse, it may be more useful to work with the number of pulses issued. They can be counted by using the VETO input as the source for GATE PULSEs, which are counted in the variable GCOUNT.

6.6.3 Count rates

Besides the live and dead times, the Pixie-4 counts the numbers of triggers in each channel, FASTPEAKS, the number of valid events with one or more channels, NUMEVENTS, and the number of valid pulses stored for each channel, NOUT. In addition, it counts the number of gate pulses for each channel, GCOUNT. FASTPEAKS and GCOUNT are inhibited when the live time is not counted. NUMEVENTS and NOUT by nature only count events captured when the live time is counted.

Count rates are then computed in the C library as follows:

Input count rate	ICR	=	FASTPEAKS / (LIVE TIME – FTDT)
Event rate	ER	=	NUMEVENTS / RUN TIME
Channel output count rate	OCR	=	NOUT / LIVE TIME
Gate count rate	GCR	=	GCOUNT / LIVE TIME

© XIA 2012. All rights reserved.

The Pixie Viewer also computes a "DAQ fraction" in Igor, defined as LIVETIME / Igor run time, which is an indication of the overall dead time lost to those processes not included in SFDT, equivalent to the last row of Table 6.3. Users are free to use the reported values to compute rates and time better matching their preferred definitions.

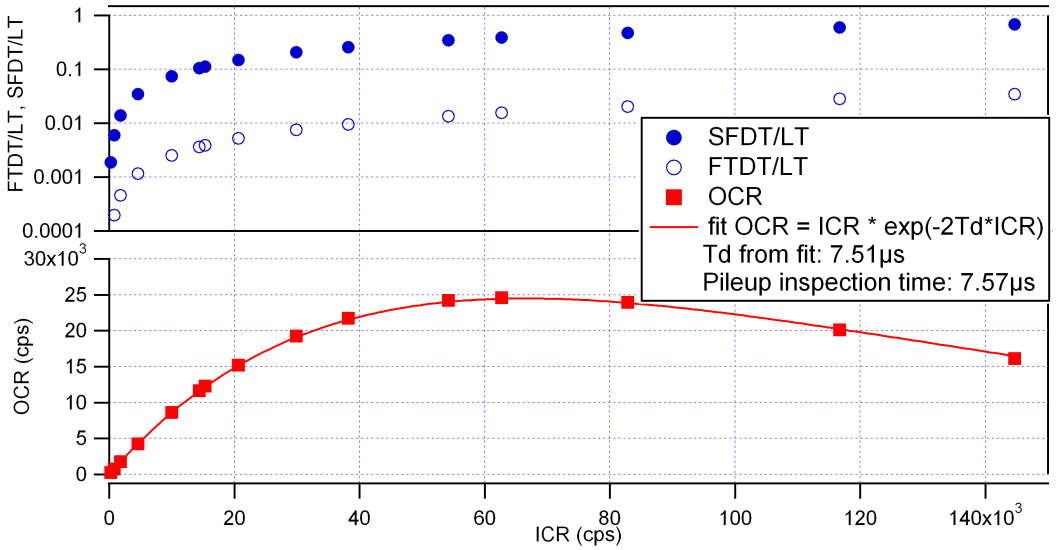


Fig. 6.13 OCR and Livetime fractions of FTDT and SFDT as a function of ICR in a Pixie-4 measurement with a random pulse generator. The measured OCR follows the expected behavior from Eq. (4) with a fit value of Td very close to the pileup inspection time (energy filter rise time plus energy filter flat top plus a few cycles).

Notes:

1) Output pulse counters are updated whenever an event has been processed; input, gate and all time counters are updated every ~7ms. Therefore reading rates at random times, e.g. clicking *Update* in the Pixie-4 Viewer, might return slight inconsistencies between input rates and output rates. At the end of the run, all rates are updated and these effects should disappear.

2) NOUT is counted for each event a channel is processed no matter if the channel had a valid hit or not. Thus a channel that is processed in "read always" mode may have an output count rate even though its input count rate is zero.

© XIA 2012. All rights reserved.

7 Operating Multiple Pixie-4 Modules Synchronously

When many Pixie-4 modules are operating as a system, it may be required to synchronize clocks and timers between them and to distribute triggers across modules. It will also be necessary to ensure that runs are started and stopped synchronously in all modules. All these signals are distributed through the PXI backplane.

7.1 Clock Distribution

In a multi-module system, there will be one clock master and a number of clock slaves or repeaters. The clock function of a module can be selected by setting jumpers JP1, JP2 and JP3 near the back of the board. Pin 2 of JP2 is the input to the board's clock distribution circuitry. It can be connected with a shunt to several other pins, thus choosing a particular clock distribution mode. The preferred clock distribution is the PXI clock mode is the chassis supports it, else the daisy-caned clock mode.

These jumpers differ slightly for modules of Revision B and Revision C/D. The clock functions themselves as described below are identical and compatible for both revisions.

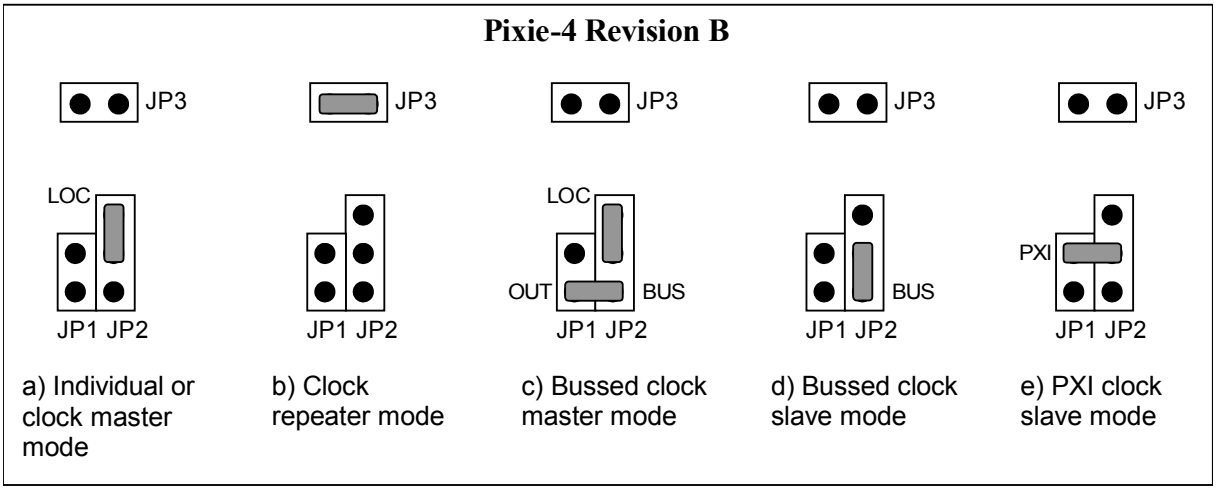


Figure 7.1. Jumper Settings for different clock distribution modes of Revision B modules. In a group of modules, there will be one daisy-chained clock master (a) in the leftmost position and several repeaters (b) OR one bussed clock master (c) and several bussed clock slaves (d). Modes (a/b) and (c/d) can not be mixed. Mode (e) can only be used with a backplane providing 37.5 MHz instead of the usual 10MHz. Jumper JP3 is located near the chip U2; it has no label.

© XIA 2012. All rights reserved.

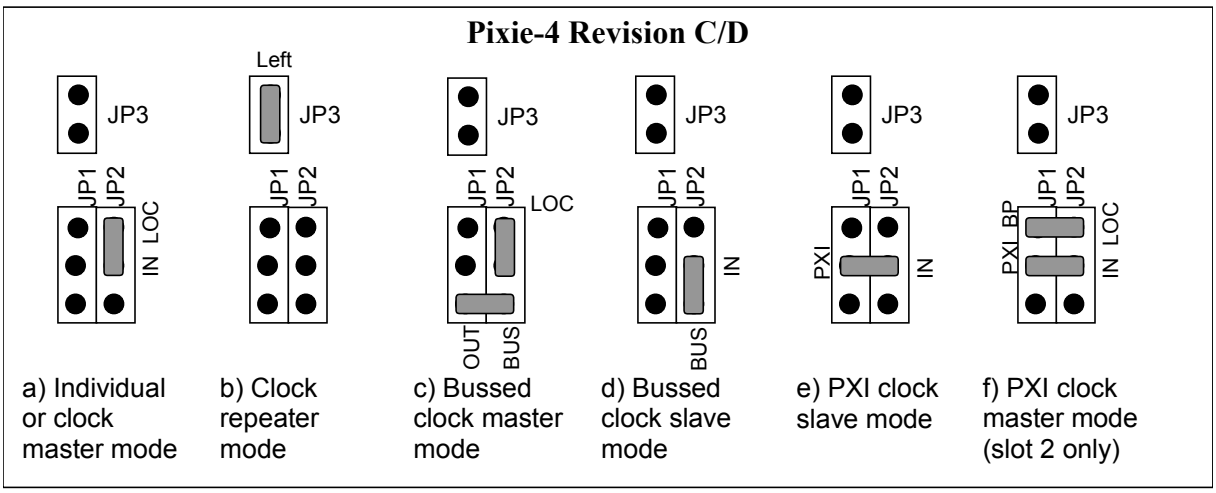


Figure 7.2. Jumper Settings for different clock distribution modes of Revision C modules. In a group of modules, there will be one daisy-chained clock master (a) in the leftmost position and several repeaters (b) OR one bussed clock master (c) and several bussed clock slaves (d) OR one PXI clock master (f) in slot 2 and several PXI clock slaves (e).
Modes (a/b), (c/d), or (e/f) can not be mixed. Mode (e) can also be used with a custom module in slot 2 or a backplane providing 37.5 MHz instead of the usual 10MHz.

7.1.1 Individual Clock mode

If only one Pixie-4 module is used in the system, or if clocks between modules do not have to be synchronized, the module should be set into individual clock mode, as shown in Figures 7.1 (a) and 7.2 (a). Connect pin 2 of JP2 (the clock input) with a shunt to pin 3 of JP2, which is labeled "LOC". This will use the on-board clock crystal as the clock source.

7.1.2 Daisy-chained Clock Mode

The preferred way to distribute clocks between modules is to daisy-chain the clocks from module to module, where each module repeats and amplifies the signal. This requires one master module, located in the leftmost slot of the group of Pixie-4 modules, with the same jumper settings as an individual module, see Figures 7.1 (a) and 7.2 (a). Configure the other modules in the chassis as clock repeaters by setting the jumpers as shown in Figures 7.1 (b) and 7.2 (b); i.e. remove all shunts from JP 1 and JP2 and set a shunt on JP3, located on top of the clock crystal U2.

Note that the clock output is always enabled, i.e. every board, independent of its clock mode, sends out a clock to its right neighbor as long as it has a clock itself. Thus make sure that no other module sits to the right of a Pixie module that uses the PXI_LBR0 line on the PXI backplane for other purposes.

7.1.3 Bussed Clock Mode

If there have to be gaps between a group of Pixie-4 modules, the daisy-chained clock distribution will not work since the chain is broken. In this case, the modules can be configured for bussed clock mode. To do so, configure one module (in any slot) as the bussed clock master as shown in Figures 7.1 (c) and 7.2 (c), i.e. set one shunt to connect pins 2 and 3 on JP2 and a second shunt to

© XIA 2012. All rights reserved.

connect Pin1 of JP1 and JP2 together("OUT" to "BUS"). The clock master will drive the clock signal to a line that is bussed to all others slots in the chassis. The drive strength is limited to 3-4 modules. All other modules should be configured as clock slaves, i.e. set a shunt to connect pin 1 and 2 of JP2 ("BUS" to clock input) as shown in Figures 7.1 (d) and 7.2 (d).

Note that the bussed clock line does usually not connect over a PCI bridge in chassis with more than 8 slots, whereas daisy-chains usually do.

Always make sure that there is no shunt on JP3, in order to disconnect from the incoming daisy-chained clock, else there will be a conflict between the two clock signals.

7.1.4 PXI Clock Mode

A further option for clock distribution is to use the PXI clock distributed on the backplane. The PXI clock is driven by default by the PXI backplane at a rate of 10 MHz - too slow to run the Pixie 4 modules. However, clock signals are individually buffered for each slot and clock skew between slots is specified to be less than 1ns, making it the preferred distribution path for sensitive timing applications. If a custom backplane provides 37.5 MHz, or if a module in slot 2 overrides the default signal from the backplane, this clock can be used as an alternative setting for Pixie-4 modules. Such a module can be a Revision C Pixie module configured as PXI clock master, shown in Figure 7.2 (f), a XIA PXI-PDM power and logic module, or any other suitable custom module.

The PXI clock master has to be configured as shown in Figure 7.2 (f). The PXI clock slaves are configured by connecting pin 2 on JP1 and JP2 together ("PXI" to clock input), as shown in Figures 7.1 (e) and 7.2 (e).

Always make sure that there is no shunt on JP3, in order to disconnect from the incoming daisy-chained clock, else there will be a conflict between the two clock signals.

7.2 Trigger Distribution

7.2.1 Trigger Distribution Within a Module

Within a module, each channel can be enabled to issue triggers. Two kinds of triggers are distributed: First, a *Fast Trigger* indicating the trigger filter just crossed the threshold, which is used to start pileup inspection and to stop the FIFOs for waveform acquisition, among other things. Second, an *Event Trigger* indicating that pileup inspection was passed, i.e. validating the event as acceptable. The Event Trigger also tells the DSP that data is ready for readout in the Trigger/Filter FPGA.

If channels are set to"group trigger" mode, each trigger enabled channel issues both kinds of triggers to the central Communication FPGA, which builds an OR of all triggers and sends it back to all channels. The channels then use the distributed fast triggers and event triggers instead of their own local triggers to capture data. In this way, one channel can cause data to be acquired at the same time in all other channels of the trigger group. The DSP then reads data from all participating channels and stores it as one event record. Each channel, trigger enabled or not, always also generates a "hit" flag if pileup inspection was passed, and DSP readout is conditional to this flag.

© XIA 2012. All rights reserved.

Even in group trigger mode, some data is captured based on the channel's local trigger. For example, the pulse height of a pulse is best determined based on the trigger from the pulse itself, not from a common group trigger that may be delayed – if several channels in a group are trigger enabled, always the *last fast trigger* before the *first event trigger* in this event is the one that counts. The following table lists which quantities are based on local or group trigger in group trigger mode. (If not in group trigger mode, all quantities are based on local triggers.)

Note that for the trigger timing to be correct, all channels in a trigger group should be set to the same energy filter rise time and flat top. Furthermore, to capture the energy of delayed channels, ensure that the coincidence window (see section 7.6) is long enough to include the maximum expected delay.

Channel	Energy	Waveform	Timestamp
Hit and trigger enabled	Based on local trigger Always read out	Based on (last) group trigger Always read out	Based on (last) group trigger unless "local trigger only" option selected Always read out
Hit, but not trigger enabled	Based on local trigger Always read out	Based on (last) group trigger Always read out	Based on (last) group trigger unless "local trigger only" option selected Always read out
Not hit (no pulse or pileup)	Based on group trigger if "estimate energy" option selected, else zero. Only read if "read always" option selected	Based on (last) group trigger Only read if "read always" option selected	Based on (last) group trigger unless "local trigger only" option selected Only read if "read always" option selected

Energy: The pulse height of a pulse is best determined based on the trigger from the pulse itself, not from a common group trigger that may be delayed. Even if a channel is not trigger enabled, energy filter values are latched some time after the local trigger filter crosses the user defined threshold. The channel is then marked as "hit" for the DSP to read out.

Even channels without "hit" are read out if the "read always" option is set for this channel. In this case, the channel's energy is usually reported as zero since there was no valid local trigger to capture the value of the energy filter. However, since the energy filter is computed continuously, setting the "estimate energy" option cause the energy filter value to be captured based on the (last) group trigger. This might be useful for channels with occasional very small pulses (below the threshold), or possibly to obtain energy estimates on piled up pulses.

Note that since the timing of the group trigger is not precise with respect to the non-triggering pulse, the energy reported is only a rough estimate. It might help to set the flat top time to a large value to make the capturing of the energy filter less time sensitive.

Waveforms: The waveforms are always captured based on the (last) group trigger for this event. Channels without "hit" are read out only if the "read always" option is set for this channel.

© XIA 2012. All rights reserved.

Timestamp: Normally, in acquisitions with shared group triggers, all channels record the (identical) timestamp of the (last) group trigger for this event. Since waveforms are captured based on the group trigger, this ensures that within a data record traces and timestamps are correlated. However, if no waveforms are recorded, there is no time-of-arrival information for possible delays between channels from the timestamps alone. In this case, setting the ”local trigger only” option preserves the time difference information by recording timestamps for each channel based on its local trigger only.

7.2.2 Trigger Distribution Between Modules

Both fast triggers and event triggers can also be distributed over the PXI backplane. Each trigger uses a common backplane line for all modules, which is set up to work as a wired-OR. Normally pulled high, the signal is driven low by the module that issues a trigger. All other modules detect the lines being low and send the triggers to all channels. In other words, the backplane line carries a system-wide trigger that essentially acts as a 5th input to the trigger OR in the Communication FPGA of each module.

Each module can be enabled to share triggers over the backplane lines or not. In this way, a trigger group can be extended over several modules or each module can form its local sub-group.

7.2.3 Trigger Distribution across PXI segment boundaries

In PXI chassis with more than 8 slots, the PCI bus as well as the PXI bussed backplane lines are divided into segments with not more than 8 slots. While the PCI bus is bridged between the segments, the PXI bussed backplane lines are usually only buffered from one segment to the next; i.e. the line in one segment drives the line in the neighboring segment. Since this buffer is essentially a one-way communication (though the direction may be selectable), no wire-OR can be build across the segment boundary. (Note: Sometimes there is no connection at all.)

For applications with more than 7 modules, the Pixie-4 have to be operated in a chained OR mode, where trigger signals are passed from module to module using the PXI nearest neighbor lines which are not interrupted by the segment boundaries. In this mode, each module ORs the trigger signal from its right neighbor with its own contributions and passes it to the left. The leftmost module issues the combined OR to a bussed PXI line. The chassis has to be configured such that the leftmost segment drives all other segments to the right. The Pixie-4 modules can be set up to operate in this mode using the chassis control panel of the Pixie Viewer. The PXI backplane buffering has to be set up with the tools provided by the chassis manufacturer: the lines named PXI_TRIG0 (fast trigger), PXI_TRIG1 (event trigger) and PXI_TRIG2 (synchronization) have to be set up to be driven from the leftmost segment.

7.2.4 Trigger Distribution between PXI chassis

In principle it is possible to distribute triggers between several chassis with Pixie-4 modules using XIA’s PXI PDM module. Please contact XIA for details.

© XIA 2012. All rights reserved.

7.2.5 External Triggers

External triggers usually do not have the correct format and fast trigger vs event trigger timing required by the Pixie-4 trigger logic. The Pixie-4 therefore includes specific logic to turn an external signal into distributed triggers.

External signals (3.3V TTL standard) can be connected to the Pixie-4 front panel input labeled "DSP-OUT". The DSP variable XETDELAY (Control field *Validation delay* ... in the CHASSIS SETUP panel controls generation of fast and event triggers. If the value is zero, no triggers are generated. If the value is nonzero, a fast trigger is issued to the backplane immediately after detection of a rising edge on the front panel, and an event trigger is issued the specified delay thereafter. As the triggers are sent to the backplane, the external triggers appear as if an additional module with a pileup inspection time (energy filter rise time plus flat top) equal to XETDELAY had seen a pulse. Sharing triggers over the backplane must be enabled even for the module connecting to the external signal.

7.3 Run Synchronization

It is possible to make all Pixie-4 modules in a system start and stop runs at the same time by using a wired-OR SYNC line on the PXI backplane. In all modules the variable SYNCHWAIT has to be set to 1. If the run synchronization is not used SYNCHWAIT must be set to 0. The variable is set by checking the corresponding checkbox in the *Run Control* tab of the Pixie Viewer.

The run synchronization works as follows. When the host computer requests a run start, the Pixie-4's DSP will first execute a run initialization sequence (clearing memory etc). At the beginning of the run initialization the DSP causes the SYNC line to be driven low. At the end of the initialization, the DSP enters a waiting loop, and allows the SYNC line to be pulled high by pullup resistors. As long as at least one of all modules is still in the initialization, the SYNC line will be low. When all modules are done with the initialization and waiting loop, the SYNC line will go high. The low->high transition will signal the DSP to break out of the loop and begin taking data.

If the timers in all modules are to be synchronized at this point, set the variable INSYNCH to 0 by checking the corresponding checkbox in the *Run* tab of the Pixie Viewer. This instructs the DSP to reset all timers to zero when coming out of the waiting loop. From then on they will remain in synch if the system is operated from one master clock.

Whenever a module encounters an end-of-run condition and stops the run it will also drive the SYNC line low. This will be detected in all other modules, and in turn stop the data acquisition.

Note that if the run synchronization is not operating properly and there was a run start request with SYNCHWAIT=1, the DSP will be caught in an infinite loop. This can be recognized by reading the variables INSYNCH and SYNCHWAIT. If after the run start request the DSP continues to show INSYNCH=0 and SYNCHWAIT=1, it is stuck in the loop waiting for an OK to begin the run. Besides rebooting, there is a software way to force the DSP to exit from that loop and to lead it back to regular operation. See the Programmer's Manual for details.

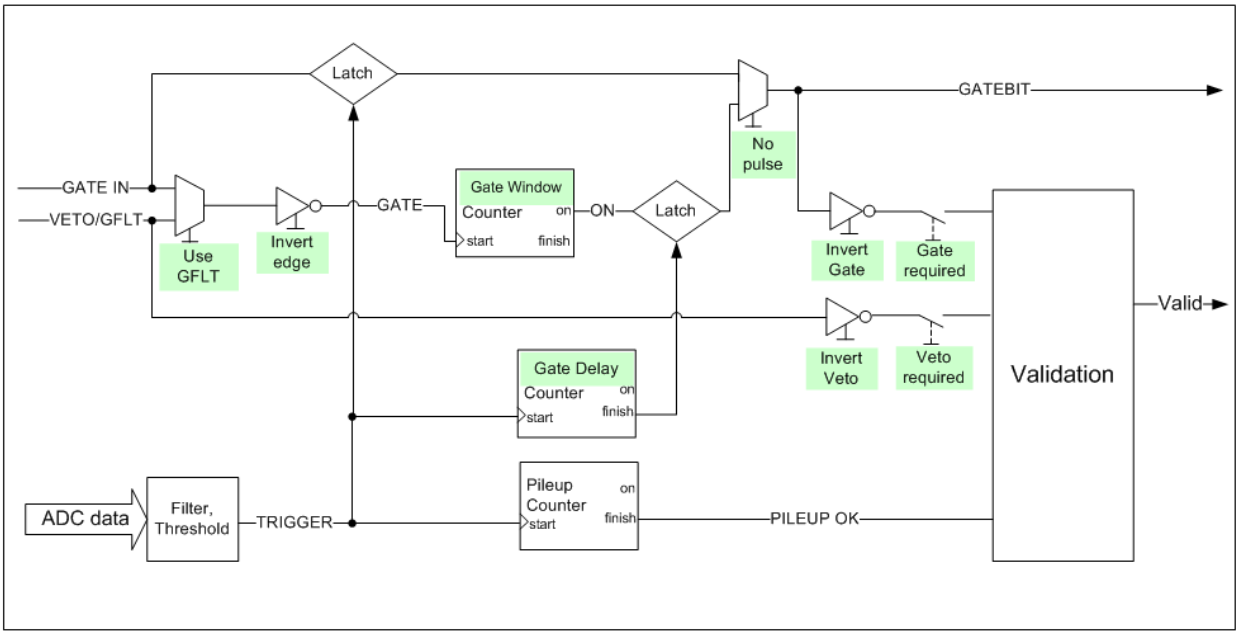
© XIA 2012. All rights reserved.

7.4 External Gate and Veto (GFLT)

It is common in larger applications to have dedicated external electronics to create event triggers or vetoes. Besides the external trigger described in section 7.2.5, the Pixie-4 also accepts a global first level trigger (GFLT). This signal acts as a validation for an event already recognized by the Pixie-4. Using multiplicities and/or other information, the external logic needs to make the decision whether to accept or reject a given event. If that decision can be made within the filter time (energy filter rise time plus flat top) of all Pixie-4 channels involved, then the GFLT function can be used. By definition, GLFT is a global signal that applies to all channels.

In a second scenario, the acquisition may have to be inhibited for certain intervals. An example is the on/off cycle of a neutron generator, and events may only be of interest if the generator is off. This condition can also be accommodated by the GFLT function, but is often described as vetoing the acquisition while the signal is on. We thus use the names GFTL and VETO interchangeably.

In a third scenario, it may be desirable to reject pulses that occur while a GATE signal is on (or off). Usually this is a dedicated signal for each channel, for example derived from a BGO shield around the detector. When the BGO shield sees a pulse, not all of the energy was deposited in the detector, and therefore this event should be rejected. The GATE signal is thus coincident with the rising edge of the detector pulse (give or take a cable delay), in contrast to the GFLT function that contributes to the event validation a filter time after the rising edge.



© XIA 2012. All rights reserved.

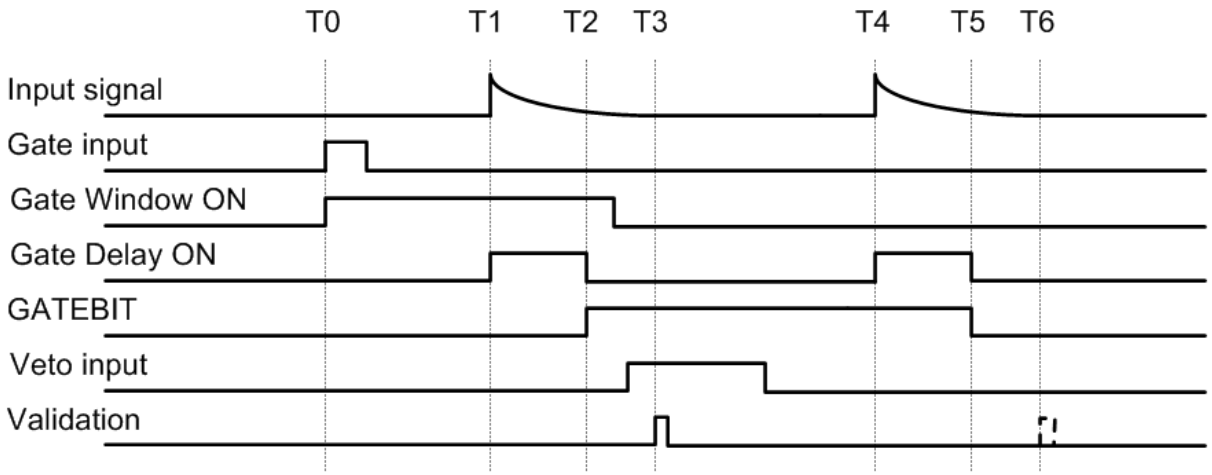


Figure 7.3. Block and timing diagrams of the GATE and VETO circuitry in the trigger/filter FPGA. If required by the user, pulses are validated only if VETO and/or GATEBIT are present during validation. GATEBIT reflects the status of the GATE input at the time of the trigger either directly or after a coincidence window is applied. Names highlighted in green are controlled by the parameters in the *Gate* tab of the PARAMETER SETUP Panel.

The Pixie-4 accommodates these scenarios in the following way (figure 7.3): In each channel, the VETO signal contributes to the validation of a pulse at time T3 if it is set by the user to be *required* to do so. The polarity of the input can be optionally *inverted*. The rising edge of the GATE signal (optionally the VETO signal, optionally *edge inverted*) starts a counter of length Gate *Window* at time T0. The time the Gate Window counter is ON is called GATE PULSE. A trigger generated at the rising edge of a pulse from the detector starts a counter of length Gate *Delay* at time T1. When the Gate *Delay* counter is finished at T2, the status of the Gate *Window* counter is latched as GATEBIT. Alternatively, the pulsing logic can be bypassed and the status of the GATE input is latched directly as GATEBIT by the trigger. If gating is *required*, the GATEBIT (optionally inverted) also contributes to the pulse validation at T3, else it is only recorded in the output data stream (in list mode data). In all cases, the trigger also starts the standard pileup counter that validates a pulse a filter time after the rising edge of the pulse. The validation thus always takes place a filter time after the rising edge of the pulse, but is optionally subject to the current status of the VETO input and/or the status of the GATE input stretched by Gate *Window* and latched a time Gate *Delay* after the rising edge of the pulse.

The action of Gate *Window* and Gate *Delay* is thus to set a coincidence window for the GATE signal, and adjust for the delay between detector signal (from the ADC) and the GATE signal. Mainly due to the pipelined processing inside the ADC, it takes about 200 ns from a rising edge at the front panel analog input of the Pixie-4 until a trigger is issued by the Pixie-4 trigger circuit. The GATE signal starting the Gate *Window* counter is therefore delayed by ~200 ns inside the FPGA to compensate for this intrinsic delay. Any delay due to cables or the physics of the experiment will be additional. Both Gate *Delay* and Gate *Window* can range from 13.3ns to 3.4 μs.

© XIA 2012. All rights reserved.

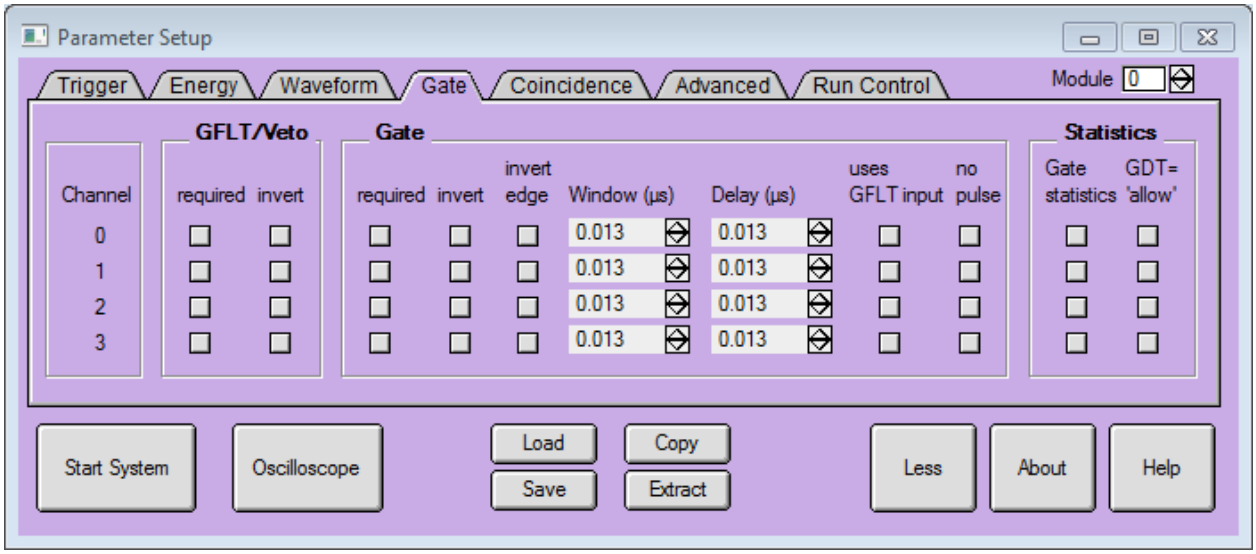


Figure 7.4. *Gate* tab of the PARAMETER SETUP Panel.

The VETO signal is distributed through the PXI backplane. Using XIA's PDM module or a custom board, external signals can be connected to the backplane. In addition, the Pixie-4 front panel input labeled "DSP-OUT" [sic] can be used to send the signal to the backplane. The input signal must be LVTTL, i.e. logic 0 = 0V, logic 1 = 3.3V. Only one module within a chassis may use this option to avoid conflicts in driving the backplane. The option is enabled in software by setting the corresponding checkbox in the Pixie Viewer's CHASSIS SETUP panel. Setting it for one module will automatically disable it for all other modules.

The GATE signal is also distributed over the backplane, using 4 PXI nearest neighbor lines. Therefore a module to the left of a Pixie-4 can be used to input 4 GATE signals to the Pixie-4. XIA's PDM can provide this function (inputs 8-5 for channel 0-3). The alternative is to use the VETO signal distributed to all modules and channels as the common GATE input for each channel.

7.5 External Status

Besides Veto, a second function for the Pixie-4's front panel input is to contribute to a wired-OR backplane line called "Status". Several modules can be enabled to contribute to the Status line. The backplane status line will be logic 1 whenever the "DSP-OUT" input of any enabled module is high (3.3V). The status of this line is read as part of the event acquisition and is stored in the list mode data. It is also possible to send the hit status bit of channel 3 to the STATUS line so that all modules will include this channel's information in their event record (see 7.6.2)

The fourth function for the Pixie-4's front panel input is to contribute one "Front" bit in the event hit pattern. If the front panel is not used as Veto or Status input, this allows recording of an externally created logic level in each module individually. For example, each module may be assigned to a detector or radiation source that is enabled/disabled individually, and so the status of that detector is recorded in the event data stream.

Notes: The front panel input can be used for Veto, Status, and Front at the same time, if necessary. The wired-OR backplane lines are of type active low, i.e. logic 1 is 0V.

© XIA 2012. All rights reserved.

7.6 Coincident Events

7.6.1 Coincidences Within a Module

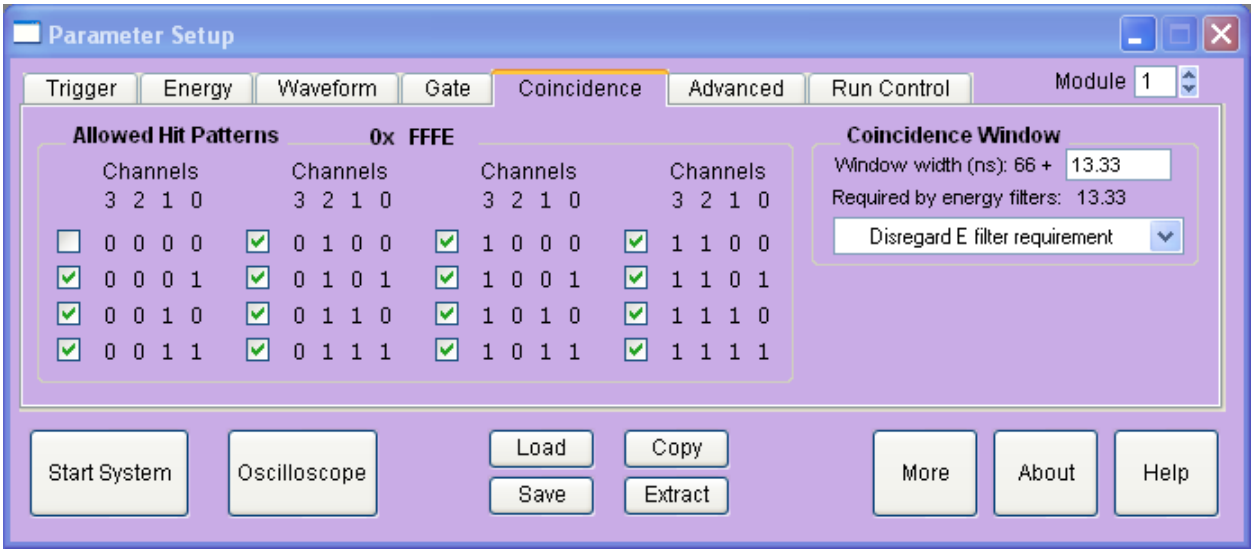


Figure 7.5. Coincidence Pattern and Coincidence Window Settings in the Pixie Viewer

In any given event, a single Pixie-4 module will have up to four channels with a "hit", i.e. a valid local pulse without pileup. The four channels thus form one of 16 possible Hit Patterns, stored in the lower 4 bits of a DSP parameter. In this representation, the Hit Pattern ranges from "no channel hit" [0000] over "only channel 1 hit" [0010] to "all four channels hit" [1111]. For each event, the Hit Pattern is checked against the user defined "Coincidence Pattern" to determine if it is acceptable (Local Test). If acceptable, the event is recorded and processed, if not, the event is rejected and data acquisition continues.

The user can define the Coincidence Pattern to accept one or more hit patterns. For example, in the *Coincidence* Tab of the PARAMETER SETUP Panel, there are 16 checkboxes for the 16 possible hit patterns, and selecting one sets the corresponding bit in the coincidence pattern. In the example shown in Figure 7.3, accepting only Hit Pattern [0001] makes the Coincidence Pattern 0x0002. Several of the check boxes can be set at the same time, for instance to accept any pattern with two or more channels. If all checkboxes are set, any possible Hit Pattern is acceptable and the Coincidence Pattern is 0xFFFF.

Each channel with a pulse above threshold, whether trigger enabled of not, contributes to the hit pattern the moment the pulse is validated as not piled up (i.e. a filter time after the rising edge of the pulse). The hit pattern is read for comparison with the coincidence pattern about 66 ns after the first pulse is validated. If several channels contribute to an event, the minimum coincidence window -- the time period in which (delayed) channels can contribute to the hit pattern -- is thus ~66 +13 ns. If longer delays between channels are expected from the physics of the experiment, this added width can be increased up to a value of ~870 microseconds (16 bit counter).

A difference in filter times between channels will cause even channels with simultaneous pulses to contribute to the hit pattern at different times. The Pixie Viewer thus calculates the required minimum window width to compensate for any such difference and displays it. The application

© XIA 2012. All rights reserved.

of this minimum value depends on the System global KEEP_CW, controlled by the popup menu below the *Window Width*. The values of the popup menu have the following meaning:

Increase for E filter, never decrease
At each change of *Window width* or the energy filter times, ensures that the *Window width* is at least the required minimum. If a larger number is entered in *Window width* or if filter times are changed which would lower the required minimum, keep the larger value.
When using this mode to modify the coincidence window as required by the experiment, one should remember to verify its value after changes in the energy filter times. In particular, when energy filter times are increased in one channel after the other, the increase in the first channel will increase the *Window width* but the increase in the last channel (making filter times equal again) will not decrease the *Window width*.

Keep at required E filter width
At all times keep the Window width at the required minimum. If a larger or smaller number is entered, it is ignored and changed back to the required minimum.
Use this mode if there are no coincidence requirements from the experiment and differences in the energy filter should be accommodated, but unnecessary large values should not linger as in the first mode.

Disregard E filter requirement
In this mode, the energy filter requirement is ignored and any value can be entered for the *Window width*. The entered value is not modified by the software.
Use this mode if channels are independent so there is no need to wait for delays between channels, or if a specified *Window width* is deemed sufficient and should not be modified by the software.

Notes:

1) Any added coincidence window width will increase the time required to process an event and thus reduce the maximum count rate.

2) In run types 0x100-0x301, pulses contributing during the readout of data (after the end of the coincidence window) are lost. The readout may take several microseconds, longer if waveforms are to be recorded.

7.6.2 Coincidences Between Modules

If more than one module is operated in the same PXI chassis, acceptance of events can also be subject to the results of a system-wide ("global") coincidence test. The result of the global test is distributed over the TOKEN backplane line. This module coincidence test takes place in the following steps:

After receiving a valid event trigger and waiting for the user defined coincidence window, each module sends its channel hit pattern to slot 2 of the chassis using the PXI STAR trigger line

Each module determines the result of the local coincidence test based on its own 4 channels. If enabled to do so, it signals the test result on the TOKEN line. If the local test passed, the TOKEN line is left pulled up (3.3V, logic 1); else the TOKEN line is driven low (logic 0).

The module in slot 2, typically XIA's PXI-PDM module, uses the up to 48 bit hit pattern from up to 12 modules (slots 3-14) to make an accept/reject decision. If the hit pattern is acceptable, the

© XIA 2012. All rights reserved.

TOKEN line is left pulled up. If not acceptable, the TOKEN line is driven low (logic 0). The decision criteria is based on a user defined control word, downloaded to the PXI-PDM by its neighboring Pixie-4 module.
The acceptance decisions implemented in the current PDM firmware are described in detail in the Pixie Viewer online help. For example, if the control word is 0x13 (0x14, 0x15, etc) events are only acceptable if at least 3, (4, 5, etc) channels are hit. If the control word is 0x0200, channel 0, but not 1, 2, and 3 must be hit in each module 0 and 1. The current firmware does not claim to cover all cases. Please contact XIA to request additional cases or to obtain verilog source code to write custom PDM firmware.

Each module, after waiting ~100ns for the global accept/reject decision to be made, captures the status of the TOKEN line and puts it in the event hit pattern. The hit pattern also contains the status of the backplane STATUS line, the result of the local coincidence test, and the status of the front panel input at this moment. Depending on user settings, the event will be recorded or discarded if the TOKEN and/or LOCAL bits are set in the hit pattern.

A full implementation of this feature thus requires an additional module in slot 2 of the chassis, receiving hit patterns over the PXI STAR Trigger lines, making a coincidence decision, and signaling the result on the TOKEN line. This can be XIA's PXI-PDM module or any other compatible PXI module. A limited coincidence decision can be made with Pixie-4 modules only, e.g. one or more "master modules" can inhibit acquisition in all other modules based on their local hit pattern.

In the Pixie Viewer, the module coincidence is configured in the CHASSIS SETUP panel (Fig. 7.4). With the checkboxes in the "Module Coincidence Setup" block, each module can be set to accept events if

a) only the local coincidence test is passed (check "*local*")

b) only the global coincidence test is passed (check "*global*")

c) either the local OR the global coincidence test is passed (check "*global*" and "*local*")

d) the global test AND local tests from all "master modules: pass (check "*global*" for all module and "*local adds to global*" for the "master modules)

Other checkboxes and controls define if a module sends its hit pattern to slot 2, if a module is writing the global coincidence control word to a neighboring PDM, and the control word to write. There is also a checkbox for each module to send the hit bit of its channel 3 to the status line. As the status line information is included in the event hit pattern in all modules, this allows one specific channel to contribute information to the event records of all modules.

© XIA 2012. All rights reserved.

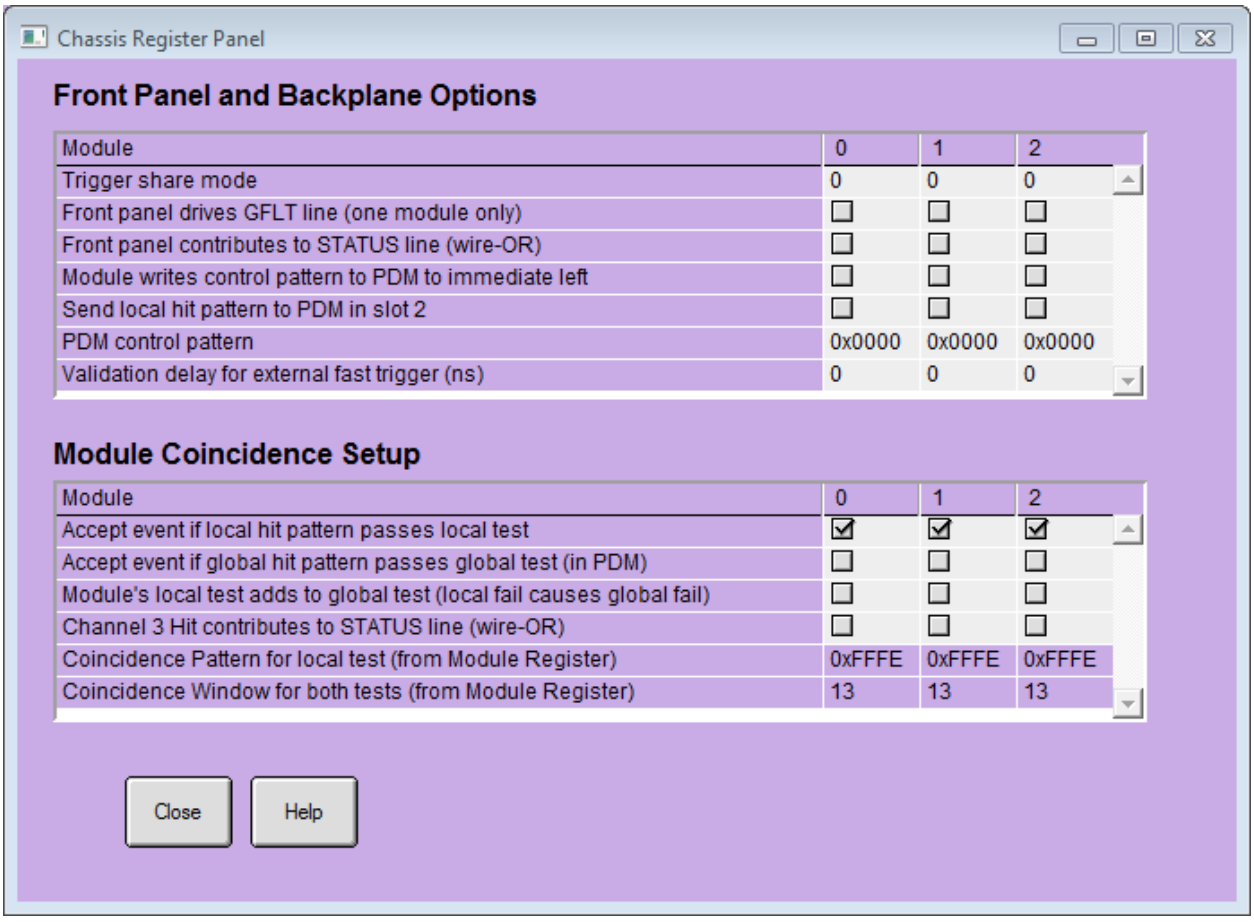

Chassis Register Panel
Front Panel and Backplane Options
Module 0 1 2
Trigger share mode 0 0 0
Front panel drives GFLT line (one module only)
Front panel contributes to STATUS line (wire-OR)
Module writes control pattern to PDM to immediate left
Send local hit pattern to PDM in slot 2
PDM control pattern 0x0000 0x0000 0x0000
Validation delay for external fast trigger (ns) 0 0 0
Module Coincidence Setup
Module 0 1 2
Accept event if local hit pattern passes local test
Accept event if global hit pattern passes global test (in PDM)
Module's local test adds to global test (local fail causes global fail)
Channel 3 Hit contributes to STATUS line (wire-OR)
Coincidence Pattern for local test (from Module Register) 0xFFFE 0xFFFE 0xFFFE
Coincidence Window for both tests (from Module Register) 13 13 13
Close
Help

Figure 7.6 Module coincidence setup in the Pixie Viewer

Examples:

1. To require a local coincidence of channels 0-1, 2-3, or both (as above), set the coincidence pattern to 0x9008 in the ModuleRegisterPanel and check only the "local test" box in the Chassis RegisterPanel
2. To require coincidence of channels 0 and 1 in Module 0, and no other channel/module matters, in the ModuleRegisterPanel set the coincidence pattern in Module 0 to 0x8888 and in all other modules to 0xFFFF. In the Chassis Register Panel, check the "global test" box for all modules and the "local adds to global" box for module 0. No PXI PDM is required.
3. To require at least 3 channels to be active in all modules, use a PDM module in slot 2 and set the [PDM control pattern] to 0x0013 for the module in slot 3. Make sure the [Module writes control pattern ...] box is checked for this module and the [Send local hit pattern to PDM] box is checked for all modules. Then check the "global test" box for all modules

© XIA 2012. All rights reserved.

8 Using Pixie-4 Modules with Clover detectors

When working with clover detectors, the Pixie-4 can be operated in a specific “clover mode”. In this mode, the DSP will calculate the pulse height for each channel as in normal operation, and in addition – for events with hits in more than one channel – calculate the sum of individual channel energies. The result, the full energy of gamma rays scattered within the clover detector, is binned in an additional “addback” spectrum.

In the current implementation of the clover mode, the spectrum length is fixed to 16K. The clover binning mode applies all runs, but in list mode runs, no sum energy is reported in the list mode data. The clover mode is enabled by setting the corresponding checkbox in the Pixie Viewer’s Module Control Register panel. There is also the option of binning only those events in the individual channel spectra that do not have multiple hits.

Additional clover functions are under development.

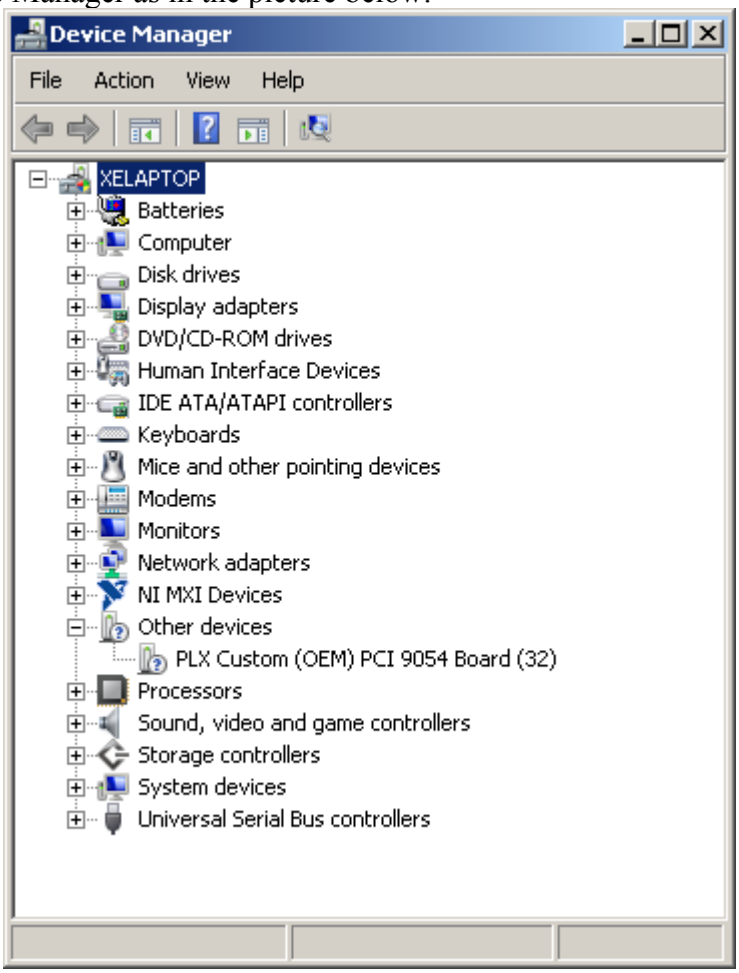
© XIA 2012. All rights reserved.

9 Troubleshooting

9.1 Startup Problems

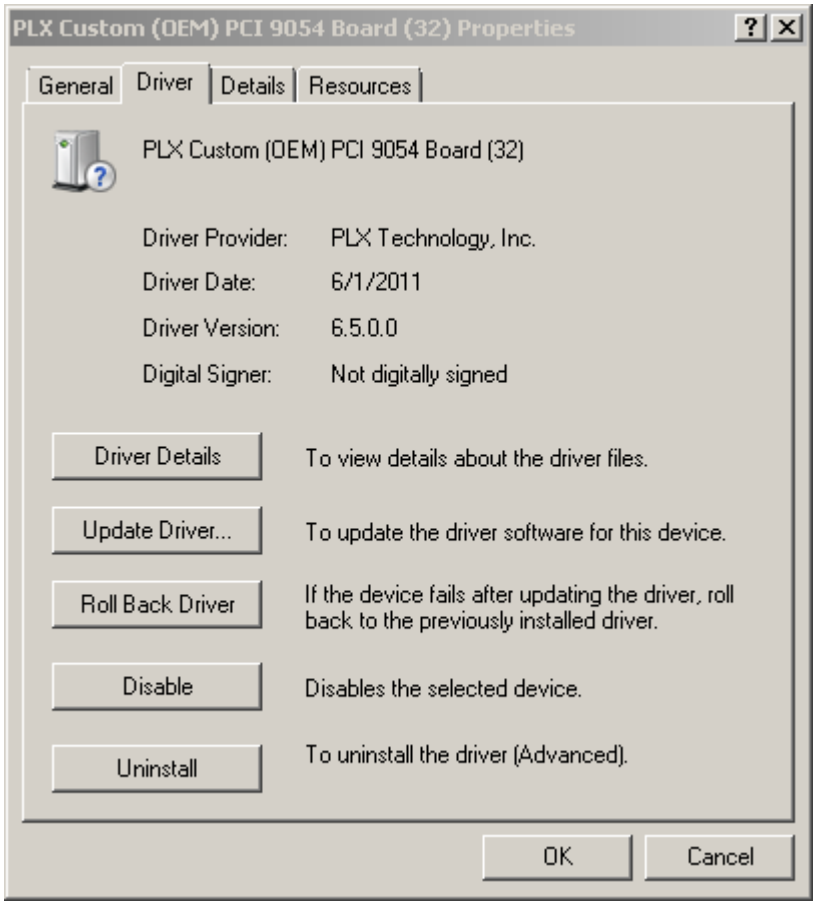
1. **Computer does not boot when Pixie module is installed in chassis**
   This is usually caused by an incorrect clock setting on the Pixie module. See section 7.1 for details. The module needs to have a valid clock to respond to the computer's scanning of the PCI bus.

2. **Computer reports new hardware found, needs driver files**
   Whenever a Pixie module is installed in a slot of the chassis for the first time, it is detected as new hardware, even if Pixie modules have been installed in other slots previously. Point Windows to the driver files provides with the software distribution. After driver installation, the module should appear in Window's Device Manager as in the picture below:



3. **Windows can not use drivers due to problem in digital signing**
   This problem seems most common in Windows 7. The PLX drivers currently only "inofficially" support Windows 7. While this is being addressed, a workaround solution is to install the complete PLX software development kit (SDK), which seems to install the PLX drivers in a way acceptable to Windows 7. The SDK is available for free from PLX (http://www.plxtech.com), but registration is required.

4. **Drivers are installed, but module does not boot**
   The Pixie-4 modules require driver version 6.5.0.0 provided by XIA, not the earlier versions (6.3.1, 5.2, 4.1 or 4.4). Modules should be listed in Window's device manager as "PLX Custom (OEM) PCI 9054 Board (32)" or " … (64)". If the "PLX" is missing, it indicates driver version 4.1 is used. A picture of the driver information reported by Windows is shown below



5. **When starting the Pixie Viewer, IGOR reports compile error or missing module**
   For IGOR to start up properly, a number of driver files have to be in the correct locations. In particular, the file "pixie.xop" has to be located in the "Igor

Extensions" folder – usually C:\Program Files\Wavemetrics\Igor Pro\Igor Extensions in a default installation – and the file "PlxApi650.dll" has to be in C:\Windows\System32.

6. **When starting up modules in the Pixie Viewer, downloads are not "successful"**
   This can have a number of reasons. Verify that
   - The files and paths point to valid locations (run the "UseHomePaths" macro)
   - The slot numbers entered in the Startup panel match the location of the modules.
   - The correct drivers are used (version 6.5.0) and modules are recognized in Window's Device Manager as shown above

7. **After starting up modules, the ADC traces show only rectangular waves.**
   This problem can be caused by downloading the wrong Communication FPGA file (e.g. a Rev. B file to a Rev. C module). Verify the file names are correct.

9.2 Acquisition Problems

1. **Signal from PMT shows unusual pulse shape**
   Verify the input jumpers are set to the correct termination. When taking the signal directly from the PMT without a preamplifier, the correct termination is usually 50Ω

2. **Missing peaks in spectra**
3. **Unusually low count rate**
4. **Unusually low Live Time**
   Open the OSCILLOSCOPE and verify that the signal is in range, i.e. that large pulses are not cut off at the upper end of the range (16K) and that the baseline is above zero

5. **Low efficiency for high energy peaks in MCA spectrum**
   At high rates, pulses overlap with the decaying tail of a previous pulse. When two or more pulses overlap in this way, higher energy pulses are more likely to go out of range
   => reduce gain and/or adjust the offset

   If the detector output shows significant ringing or overshoots, it can happen that the Pixie-4 triggers twice on the same pulse (first on the rising edge, then on the overshoot). This would be more likely for higher energy pulses, because the ringing or overshoot has a larger amplitude.
   => increase the trigger threshold and/or the trigger filter rise time or use the advanced options to "pause" or (for low count rates) disable the pileup inspection.

6. **Data collection in list mode has low DAQ fraction**
7. **SFDT is a large fraction of the live time**
8. **Rate at which list mode data is written to file is low**

   The number of events collected in a given time depends on a) the data per event, b) time required to record an event, and c) the data transfer rate.

   To reduce a),

   - run in compressed list mode (run types 0x101-103)
   - shorten the tracelength as much as possible (even in compressed list mode!)
   - remove the "read always" and "good channel" option for unused channels

   To reduce b)

   - reduce the coincidence window to the minimum possible
   - if no pulse shape analysis is required in compressed list mode runs, set the tracelength to zero
   - do not require pulse shape analysis

   To increase c)

   - run in 32x buffer or 16/16 double buffer mode
   - avoid frequent updates of run statistics and spectra
   - set the polling time to a small value (0.1-0.01)
   - verify the number of events/buffer is set to the maximum

9. **Bad energy resolution in MCA spectrum**

   - verify the decay time is set correctly
   - increase energy filter rise time
   - make the energy filter flat top approximately equal to the rise time of the pulse
   - ensure the "integrator" is set to zero
   - if "integrator" is set to 1 on purpose (e.g. fast scintillator pulses), make sure the energy filter flat top covers the entire pulse
   - if "integrator" is set to 2 on purpose (e.g. square pulses), make sure the energy filter flat top covers the portion of the pulse that should be disregarded for the energy measurement (e.g. the rising edge)

10 Appendix A

This section contains hardware-related information.

10.1 Front end jumpers for termination and attenuation

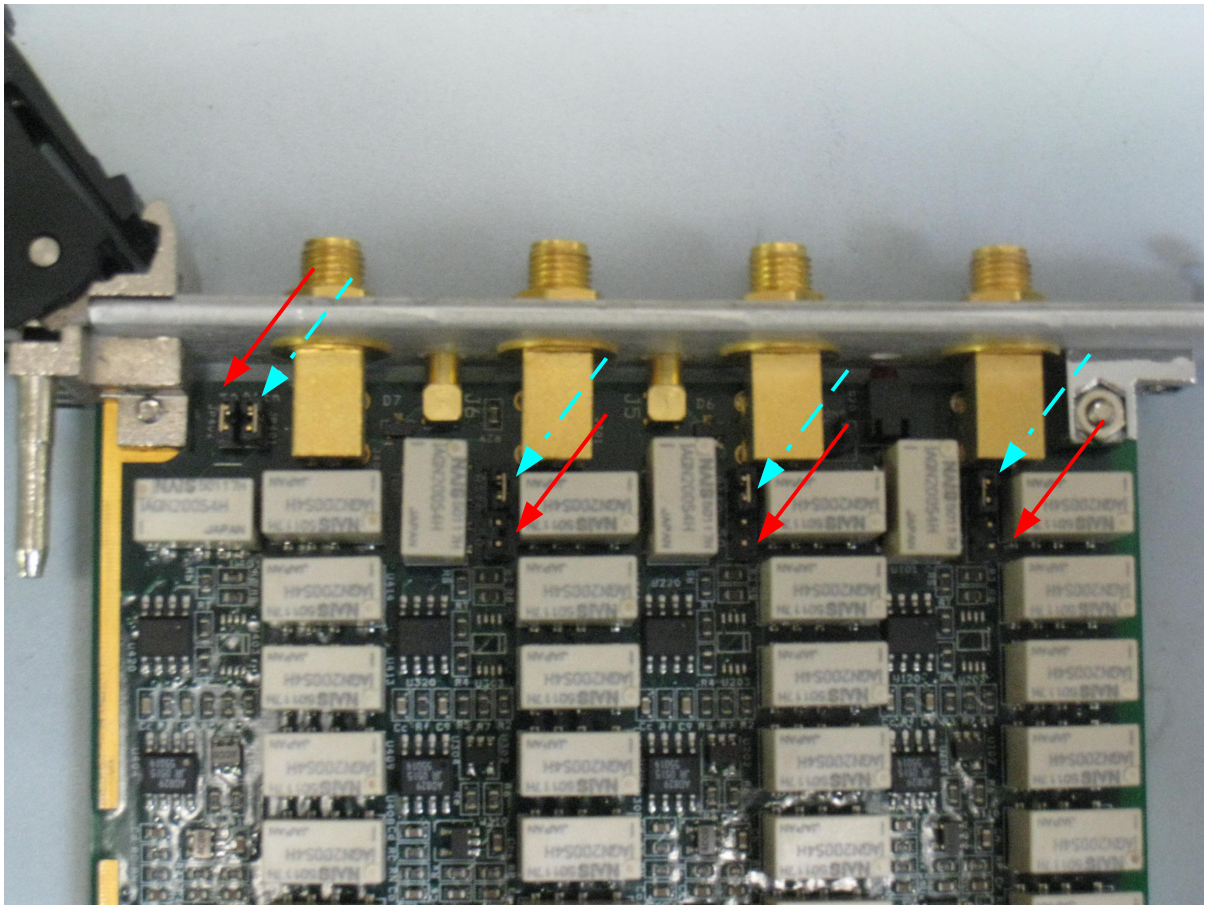


Table 10.1: Analog conditioning selection jumpers on Pixie-4 modules. x=1..4 for channel 0..3. Jumpers are marked with solid red (50Ω) and dashed blue (attenuation) arrows.

Jumper reference	PCB Label	Function
JPx01	"ATTN"	Remove only if you require attenuation. Attenuation will be 1:7.5 if JPx02 is set.
JPx02	"50"	Set for input impedance of 50Ω. If not set, input impedance is 5KΩ.
JPx05		(Revision B only) Set to "VGA" to enable contributions of the variable gain amplifier to the overall system gain for fine tuning of the gain.

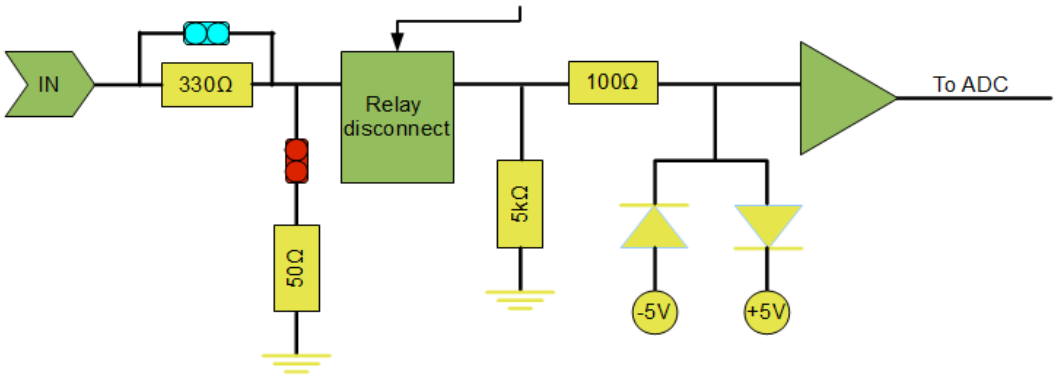


Figure 10.1. Simplified input stage of Pixie-4 showing jumpers, input termination and attenuation, and the overvoltage protection circuit.

10.2 Clock Jumpers

Table 10.2: On-board jumper settings for the clock distribution on Pixie-4 modules.

Clock mode	JP1 and JP2	JP3	PCB Label
Single Module	Connect pins 2 and 3 of JP2	not set	LOC to IN
Daisy-Chained Clock Master	Connect pins 2 and 3 of JP2	not set	LOC to IN
Daisy-Chained Clock Repeater	Not set	set	Left
Bussed Clock Master	Connect pins 2 and 3 of JP2 Connect pin1, JP1 to pin 1, JP2	not set	LOC to IN OUT to BUS
Bussed Clock Slave	Connect pins 1 and 2 of JP2	not set	BUS to IN
Clock Slave with PXI clock	Connect pin2, JP1 to pin 2, JP2	not set	PXI to IN
Clock Master for PXI clock (Revision C only)	Connect pin2, JP1 to pin 2, JP2 Connect pin3, JP1 to pin 3, JP2	not set	PXI to IN LOC to BP

10.3 PXI backplane pin functions

Table 10.3: Pins of the J2 backplane connector defined in the PXI standard used by the Pixie-4. Pins not listed are not connected except for pull-ups to 5V recommended by the PXI standard.

J2 pin number	PXI pin name	Connection Type	Pixie pin function
1A	LBL9	Left neighbor	Event Trigger output (chained OR)
3A	LBR7	Right neighbor	reserved
16A	TRIG1	Bussed	Event Trigger
17A	TRIG2	Bussed	Veto
18A	TRIG3	Bussed	Sync
19A	LBL2	Left neighbor	Sync output (chained OR)
20A	LBR4	Right neighbor	reserved
21A	LBR0	Right neighbor	Clock output
16B	TRIG0	Bussed	Fast Trigger
18B	TRIG4	Bussed	Status
20B	LBR5	Right neighbor	reserved
1C	LBL10	Left neighbor	Fast Trigger output (chained OR)
3C	LBR8	Right neighbor	reserved
18C	TRIG5	Bussed	Token
19C	LBL3	Left neighbor	Control data to PDM (left)
20C	LBL0	Left neighbor	Clock input
2D	LBL7	Left neighbor	GATE input channel 3
3D	LBR9	Right neighbor	Event Trigger input (chained OR)
15D	LBL6	Left neighbor	GATE input channel 2
17D	STAR	Star trigger to slot 2	Hit pattern to slot 2
19D	LBL4	Left neighbor	GATE input channel 0
21D	LBR2	Right neighbor	Sync input (chained OR)
2E	LBL8	Left neighbor	reserved
3E	LBR10	Right neighbor	Fast Trigger input (chained OR)
15E	LBR6	Right neighbor	reserved
16E	TRIG7	Bussed	Bussed Clock
17E	CLK10	Clock	PXI Clock
19E	LBL5	Left neighbor	GATE input channel 1
21E	LBR3	Right neighbor	reserved

10.4 Control and Status Register Bits ("CSR")

Table 10.4: Control and Status Register of the Pixie-4 System FPGA

0x0001	Bit 0	RunEna	Set to 1 to start data acquisition or 0 to stop. Automatically cleared when DSP de-asserts Active to end run.
0x0002	Bit 1	Unused	Reserved for future use.
0x0004	Bit 2	PCIactive	Set to reserve external memory I/O for host
0x0008	Bit 3	Unused	Reserved for future use.
0x0010	Bit 4	DSPReset	Write only. Set to reset DSP processor to initiate program download
0x0020	Bit 5	SynchCtrl	Read only. If low, module is busy with run initialization, has filled its I/O buffer with data, or is finished with the run.
0x0040	Bit 6	Unused	Reserved for future use.
0x0080	Bit 7	Unused	Reserved for future use.
0x0100	Bit 8	SynchFlag	Read only. Reserved for future use.
0x0200	Bit 9	Live*	Read only. If zero, DSP is taking data.
0x0400	Bit 10	Unused	Reserved for future use.
0x0800	Bit 11	Unused	Reserved for future use.
0x1000	Bit 12	Unused	Reserved for future use.
0x2000	Bit 13	Active	Read only. If set, there is a run in progress.
0x4000	Bit 14	LAMState	Read only. If set, LAM is set internally.
0x8000	Bit 15	Unused	Reserved for future use.